МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра **САП**

ЗВІТ

до виконання лабораторної роботи №2

На тему:” Вивчення роботи переривань, ШІМ та АЦП програмованого мікроконтролера Arduino”

з курсу “Мікропроцесорні системи ”

Варіант - 4

Виконав:

Студент гр.ПП-31

Гаврилюк Н. О.

Прийняв:

Доцент кафедри САП

Головатий А.І.

ЛЬВІВ - 2025

Мета: дослідити можливості створення мелодії, за допомогою спеціальних бібліотек, в Arduino; дослідити роботу переривань, АЦП та ШІМ в контролерах Arduino.

Теоретичні відомості

Для відтворення звукового сигналу, використовуючи мікроконтролер Arduino, використовують команду *tone()*, яка генерує на виводі мікроконтролера прямокутний сигнал заданої частоти (з коефіцієнтом заповнення 50%). Функція також дозволяє задавати тривалість сигналу. Однак, якщо тривалість сигналу не вказана, він буде генеруватися доти, поки не буде викликана функція *noTone()*. Для відтворення звуку порт Arduino можна підключити до зумеру або динаміку.

У кожен момент часу може генеруватися тільки один сигнал заданої частоти. Якщо сигнал вже генерується на будь-якому виводі, то використання функції *tone()* для цього виводу призведе до зміни частоти цього сигналу. У той же час виклик функції *tone()* для іншого виводу не матиме ніякого ефекту. Для відтворення різних звуків на кількох виводах, необхідно спершу викликати *noTone()* на одному і тільки після цього використовувати функцію *tone()* на наступному виводі.

tone (pin, frequency) / tone (pin, frequency, duration).Параметри:

- pin: вивід, на якому буде генеруватися сигнал;
- frequency: частота сигналу в Герцах (unsigned int);
- duration: тривалість сигналу в мілісекундах (unsigned long);
- значення, що повертає. немає.

Приклад формування тонального сигналу наведений у додатку В (програма Tone).

Переривання. це сигнали, що переривають нормальний перебіг програми. Вони використовуються для апаратних пристроїв, що вимагають негайної реакції на появу подій. Обробка переривань у мікроконтролері відбувається за допомогою модуля переривань, який приймає запити переривання й організовує перехід до виконання визначеної програми. Запити переривання можуть надходити як від зовнішніх джерел, так і від джерел, розташованих у різних внутрішніх модулях мікроконтролера. Тактові кнопки досить часто підключають до входів зовнішніх переривань (0. pin D2, 1. pin D3). Для роботи з зовнішніми перериваннями використовують функції Arduino *attachInterrupt (interrupt, function, mode), detachInterrupt (pin), interrupts ()*.

Для роботи з АЦП та ШІМ використовуються аналогові функції портів A0-A7 Arduino Nano: *analogReference (type), analogRead (pin)*, *analogWrite (pin, value)*. Більш детально ці функції описані в розділі 2.5 [13, 14].

У додатку В наводиться програма LED3, яка демонструє роботу з зовнішнім перериванням та LED4, яка використовує формування ШІМ сигналу для зміни яскравості світіння світлодіода. У програмі LED5 зчитується значення аналогової напруги з RV1. Це значення використовується для формування частоти мигання світлодіода та зміни його яскравості.

Хід виконання роботи

1. Підключити схему до комп'ютера через USB порт плати Arduino та/або запустити віртуальний стенд у середовищі Proteus 8.

2. Завантажити програму Tone (додаток В) до лабораторного макета / віртуального стенду, попередньо виконати з'єднання елементу «buzzer» та тактових кнопок S1-S4 у відповідність до програми. Дослідити роботу програми.

Завантажив програму TONE та запрограмував Arduino:

```
int buttonState = 0; void setup()
{
pinMode(2, INPUT_PULLUP);
pinMode(3, OUTPUT);
}
void loop()
{
buttonState = digitalRead(2); if (buttonState == LOW)
{
//tone(pin, frequency, duration) tone(3, 923, 300);
delay(300); tone(3, 323, 300);
delay(300); tone(3, 523, 300);
delay(300);
}
else
{
noTone(3);
}
}
```

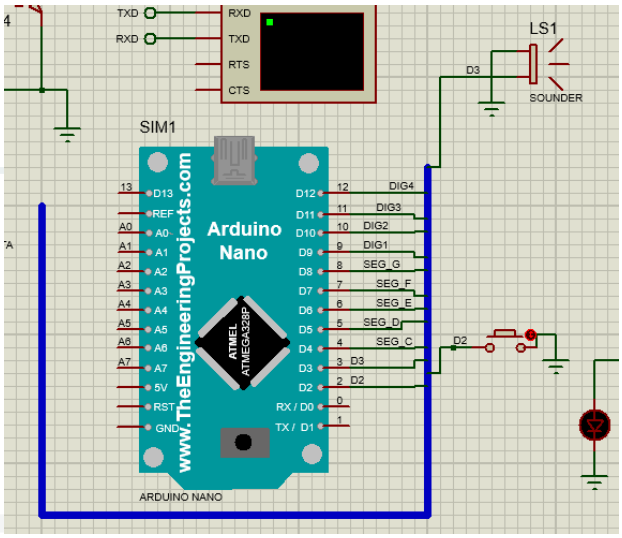

Рис.1-2. Виконання завдання

3. Завантажити програму LED3 (додаток В) до лабораторного макета / віртуального стенду, попередньо виконати з'єднання HL1-HL6 та тактових кнопок S1-S4 у відповідність до програми. Дослідити роботу програми.

```ino
// Interupt INT1 (D3)
// з'єднати D3 з будь-яким S1-S4

int led = 5;
volatile int state = LOW;

void setup(){
 pinMode(led, OUTPUT);
 attachInterrupt(1, blink, CHANGE);
  blink() ;
}

void loop(){ digitalWrite(led, state);
}

void blink(){ state = !state;

}

```

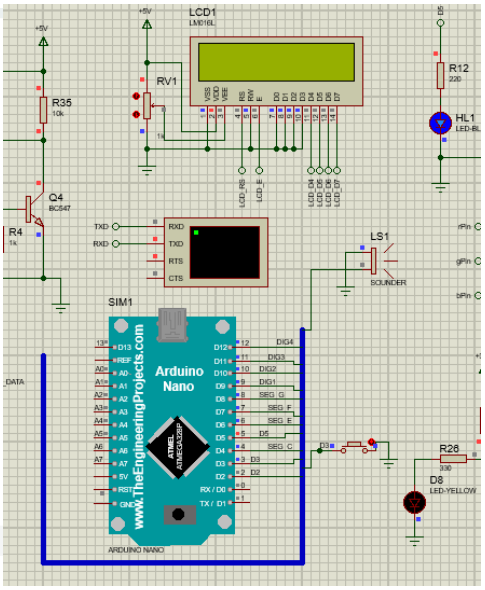

Рис.3-4. Виконання завдання

4. Завантажити програму LED4 (додаток В) до лабораторного макета / віртуального стенду, попередньо виконати з’єднання HL1-HL6 та тактових кнопок S1-S4 у відповідність до програми. Дослідити роботу програми.

```ino
#define LED_PIN 6
void setup()
{
pinMode(LED_PIN, OUTPUT);
}

void loop()
{
analogWrite(LED_PIN, 85);
 delay(250);

analogWrite(LED_PIN, 170);
 delay(250);

analogWrite(LED_PIN, 255);
 delay(250);
}

```

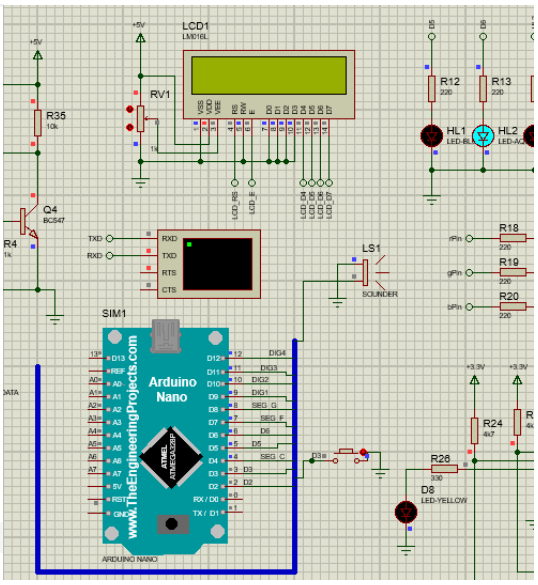

Рис.5-6. Виконання завдання

5. Завантажити програму LED5 (додаток В) до лабораторного макета / віртуального стенду, попередньо виконати з’єднання HL1-HL6 та тактових кнопок S1-S4 у відповідність до програми. Дослідити роботу програми.

```
1  void setup() {
2    pinMode(11, OUTPUT);
3    pinMode(12, OUTPUT);
4    pinMode(A1, INPUT);
5    Serial.begin(9600);
6  }
7
8  void loop() {
9    int val1 = analogRead(A1);
10   Serial.println(val1);
11   int val2 = val1 / 4;
12   analogWrite(11, val2);
13   digitalWrite(12, HIGH);
14   delay(val1);
15   digitalWrite(12, LOW);
16   delay(val1);
17 }
18
```

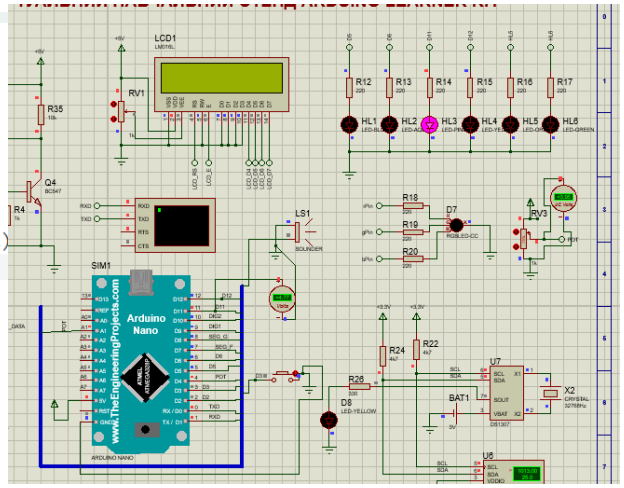

Рис.7-8. Виконання завдання

Індивідуальне завдання

Реалізувати програму, у якій кожній кнопці призначена своя нота або мелодія.

Код в Arduino Ide:

```
void setup() {
 pinMode(4,INPUT);
 pinMode(5,INPUT);
 pinMode(6,INPUT);
 pinMode(7,INPUT);
 pinMode(3,OUTPUT);
}

void loop()
{
  if(digitalRead(4)==LOW)
  {
    tone(3,150);
  }
  else if(digitalRead(5)==LOW){
    tone(3,300);
  }
  else if(digitalRead(6)==LOW){
    tone(3,450);
  }
```

```
  else if(digitalRead(7)==LOW){
    tone(3,600);
  }
  else{
    noTone(3);
  }

}
```

Код для МК AVR:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define BUZZER PB3  // Вихід під зумер
#define BTN1 PD4
#define BTN2 PD5
#define BTN3 PD6
#define BTN4 PD7

void buzzer_tone(uint16_t freq) {
    if(freq == 0) {
        // Вимкнути звук
        TCCR0 = 0;
        PORTB &= ~(1<<BUZZER);
        return;
    }
    // Налаштування таймера 0 для генерації частоти
    uint16_t ocr = (F_CPU/(2*freq*64)) - 1;
    TCCR0 = (1<<WGM01) | (1<<WGM00) | (1<<COM00) | (1<<CS01) | (1<<CS00);
// Fast PWM, prescaler 64
    OCR0 = ocr;
}
```

```
int main(void) {
    // Налаштування кнопок як входи з підтяжкою
    DDRD &= ~((1<<BTN1)|(1<<BTN2)|(1<<BTN3)|(1<<BTN4));
    PORTD |= (1<<BTN1)|(1<<BTN2)|(1<<BTN3)|(1<<BTN4); // внутрішня підтяжка

    // Зумер як вихід
    DDRB |= (1<<BUZZER);

    while(1) {
        if(!(PIND & (1<<BTN1))) {
            buzzer_tone(150);
        } else if(!(PIND & (1<<BTN2))) {
            buzzer_tone(300);
        } else if(!(PIND & (1<<BTN3))) {
            buzzer_tone(450);
        } else if(!(PIND & (1<<BTN4))) {
            buzzer_tone(600);
        } else {
            buzzer_tone(0);
        }
    }
}
```

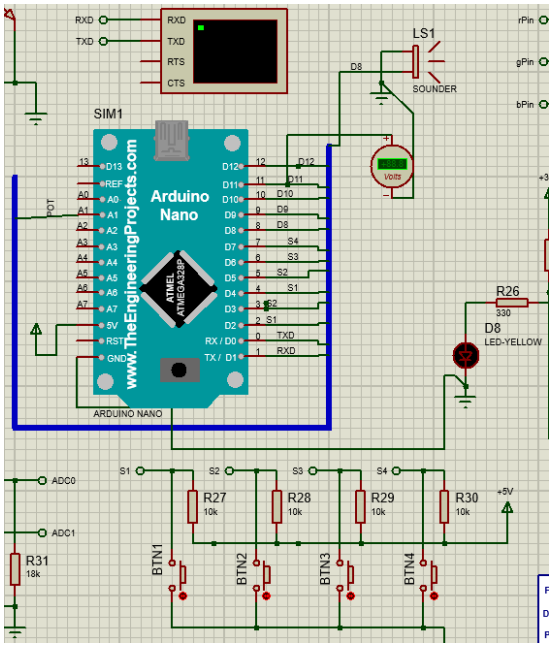

Рис.9. Виконання завдання

Реалізувати програму, у якій тональність звукового сигналу встановлюється потенціометром RV1.

Код:

```
void setup() {
 pinMode(A1,INPUT);
 pinMode(8,OUTPUT);
 Serial.begin(9600);
}

void loop()
{
  tone(8,analogRead(A1));
  Serial.println(analogRead(A1));
}
```

Код для МК AVR:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define BUZZER PB0   // Вихід під зумер
#define F_CPU 16000000UL
```

```
void ADC_init() {
   ADMUX = (1<<REFS0); // AREF = AVCC, канал ADC0 (можно змінити на A1 -> MUX0=1)
   ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0); // вкл. ADC, prescaler 128
}

uint16_t ADC_read(uint8_t channel) {
   ADMUX = (ADMUX & 0xF0) | (channel & 0x0F);
   ADCSRA |= (1<<ADSC); // старт конверсії
   while(ADCSRA & (1<<ADSC)); // чекаємо завершення
   return ADC;
}

void buzzer_tone(uint16_t freq) {
   if(freq == 0) {
      TCCR0 = 0;
      PORTB &= ~(1<<BUZZER);
      return;
   }
   uint16_t ocr = (F_CPU/(2*freq*64)) - 1;
   TCCR0 = (1<<WGM01)|(1<<WGM00)|(1<<COM00)|(1<<CS01)|(1<<CS00); // Fast PWM, prescaler 64
   OCR0 = ocr;
}

void UART_init(uint16_t baud) {
   uint16_t ubrr = F_CPU/16/baud-1;
   UBRRH = (ubrr>>8);
   UBRRL = ubrr;
   UCSRB = (1<<TXEN); // вкл. передачу
   UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); // 8 біт даних
```

```
}

void UART_send(uint16_t value) {
    char buffer[10];
    itoa(value, buffer, 10);
    for(int i=0; buffer[i]; i++){
        while(!(UCSRA & (1<<UDRE)));
        UDR = buffer[i];
    }
    while(!(UCSRA & (1<<UDRE)));
    UDR = '\n';
}

int main(void) {
    DDRB |= (1<<BUZZER); // зумер як вихід
    ADC_init();
    UART_init(9600);

    while(1) {
        uint16_t val = ADC_read(1); // читаємо A1
        buzzer_tone(val);
        UART_send(val);
        _delay_ms(100);
    }
}
```

Реалізувати на світлодіодах HL1-HL6 програму, у якій кожний з них світиться з різною яскравістю (використовуються порти D3, D5, D6, D9, D10, D11).

Код в Arduino Ide:

```
void setup() {
 pinMode(3,OUTPUT);
 pinMode(5,OUTPUT);
 pinMode(6,OUTPUT);
 pinMode(9,OUTPUT);
```

```
 pinMode(10,OUTPUT);
 pinMode(11,OUTPUT);
}

void loop() {
  analogWrite(3,0);
  analogWrite(5,51);
  analogWrite(6,102);
  analogWrite(9,153);
  analogWrite(10,204);
  analogWrite(11,255);
  delay(10000);
}
```

Код для МК AVR:

```
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
    // Встановлюємо порти як виходи
    DDRD |= (1<<PD3)|(1<<PD5)|(1<<PD6); // OC2 PD3, PD5, PD6 (Timer0 і Timer2)
    DDRB |= (1<<PB1)|(1<<PB2)|(1<<PB3); // OC1A, OC1B, інші (Timer1)

    // Налаштування Timer0 для PD5 (OC0) і PD6 (OC0)
    // Fast PWM, non-inverting
    TCCR0 = (1<<WGM00)|(1<<WGM01)|(1<<COM01)|(1<<CS01); // prescaler=8

    // Налаштування Timer1 для PB1 (OC1A) і PB2 (OC1B)
    TCCR1A = (1<<WGM10)|(1<<COM1A1)|(1<<COM1B1); // 8-bit Fast PWM
    TCCR1B = (1<<WGM12)|(1<<CS11); // prescaler=8

    // Налаштування Timer2 для PD3 (OC2)
    TCCR2 = (1<<WGM20)|(1<<WGM21)|(1<<COM21)|(1<<CS21); // prescaler=8
```

```
    while(1) {
        // Встановлюємо ШІМ значення (0-255)
        OCR0 = 51;   // PD5
        OCR2 = 102;  // PD3
        OCR1AL = 153; // PB1
        OCR1BL = 204; // PB2
        OCR0 = 255;   // PD6
        _delay_ms(10000); // 10 секунд
    }
}
```

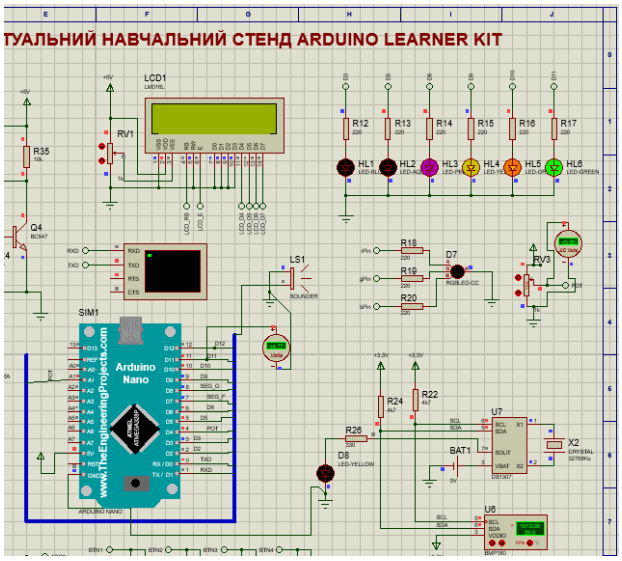

Рис. 10. Виконання завдання

Реалізувати на світлодіодах HL1-HL6 програму (використовуються порти D3, D5, D6, D9, D10, D11), у якій кнопками S1/S4 збільшується/зменшується яскравість їх світіння.

Код:

```
void setup() {
 pinMode(4,INPUT);
 pinMode(7,INPUT);
 pinMode(3, OUTPUT);
 pinMode(5, OUTPUT);
```

```
 pinMode(6, OUTPUT);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);
 pinMode(11, OUTPUT);
 Serial.begin(9600);
}
int a=0;
void loop()
{
  if(digitalRead(4)==LOW)
  {
    if(a<255)
    {
      a++;
    }
  }
  if(digitalRead(7)==LOW)
  {
    if(a>0)
    {
      a--;
    }
  }
  Serial.println(a);
  analogWrite(3,a);
  analogWrite(5,a);
  analogWrite(6,a);
  analogWrite(9,a);
  analogWrite(10,a);
  analogWrite(11,a);
}
```

Код для МК AVR:

```
#include <avr/io.h>
```

```
#include <util/delay.h>
#include <avr/interrupt.h>

#define BTN_UP PD4
#define BTN_DOWN PD7
#define F_CPU 16000000UL

volatile uint8_t a = 0;

// Ініціалізація ADC, якщо потрібно (тут не використовується)
// Ініціалізація UART
void UART_init(uint16_t baud) {
   uint16_t ubrr = F_CPU/16/baud-1;
   UBRRH = (ubrr>>8);
   UBRRL = ubrr;
   UCSRB = (1<<TXEN); // ввімкнути передачу
   UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0); // 8 біт даних
}

void UART_send(uint8_t val) {
   char buffer[4];
   itoa(val, buffer, 10);
   for(int i=0; buffer[i]; i++) {
      while(!(UCSRA & (1<<UDRE)));
      UDR = buffer[i];
   }
   while(!(UCSRA & (1<<UDRE)));
   UDR = '\n';
}

// Ініціалізація PWM для 6 виходів (Timer0, Timer1, Timer2)
void PWM_init(void) {
```

```
    DDRB |= (1<<PB3)|(1<<PB1)|(1<<PB2); // OC0 (PB3), OC1A (PB1), OC1B (PB2)
    DDRD |= (1<<PD3)|(1<<PD5)|(1<<PD6); // OC2 (PD3), PD5, PD6

    // Timer0 Fast PWM non-inverting
    TCCR0 = (1<<WGM00)|(1<<WGM01)|(1<<COM01)|(1<<CS01);

    // Timer1 8-bit Fast PWM non-inverting
    TCCR1A = (1<<WGM10)|(1<<COM1A1)|(1<<COM1B1);
    TCCR1B = (1<<WGM12)|(1<<CS11);

    // Timer2 Fast PWM non-inverting
    TCCR2 = (1<<WGM20)|(1<<WGM21)|(1<<COM21)|(1<<CS21);
}

// Встановлення ШІМ значень
void set_PWM(uint8_t val) {
    OCR0 = val;       // Timer0 -> PB3
    OCR1AL = val;     // Timer1A -> PB1
    OCR1BL = val;     // Timer1B -> PB2
    OCR2 = val;       // Timer2 -> PD3
    // решта пінів під PWM треба налаштувати за схемою
}

int main(void) {
    // Налаштування кнопок як входи з підтяжкою
    DDRD &= ~((1<<BTN_UP)|(1<<BTN_DOWN));
    PORTD |= (1<<BTN_UP)|(1<<BTN_DOWN);

    PWM_init();
    UART_init(9600);

    while(1) {
```

```
        if(!(PIND & (1<<BTN_UP))) {
            if(a<255) a++;
        }
        if(!(PIND & (1<<BTN_DOWN))) {
            if(a>0) a--;
        }

        set_PWM(a);
        UART_send(a);
        _delay_ms(100); // антидребезг та зручна швидкість UART
    }
}
```

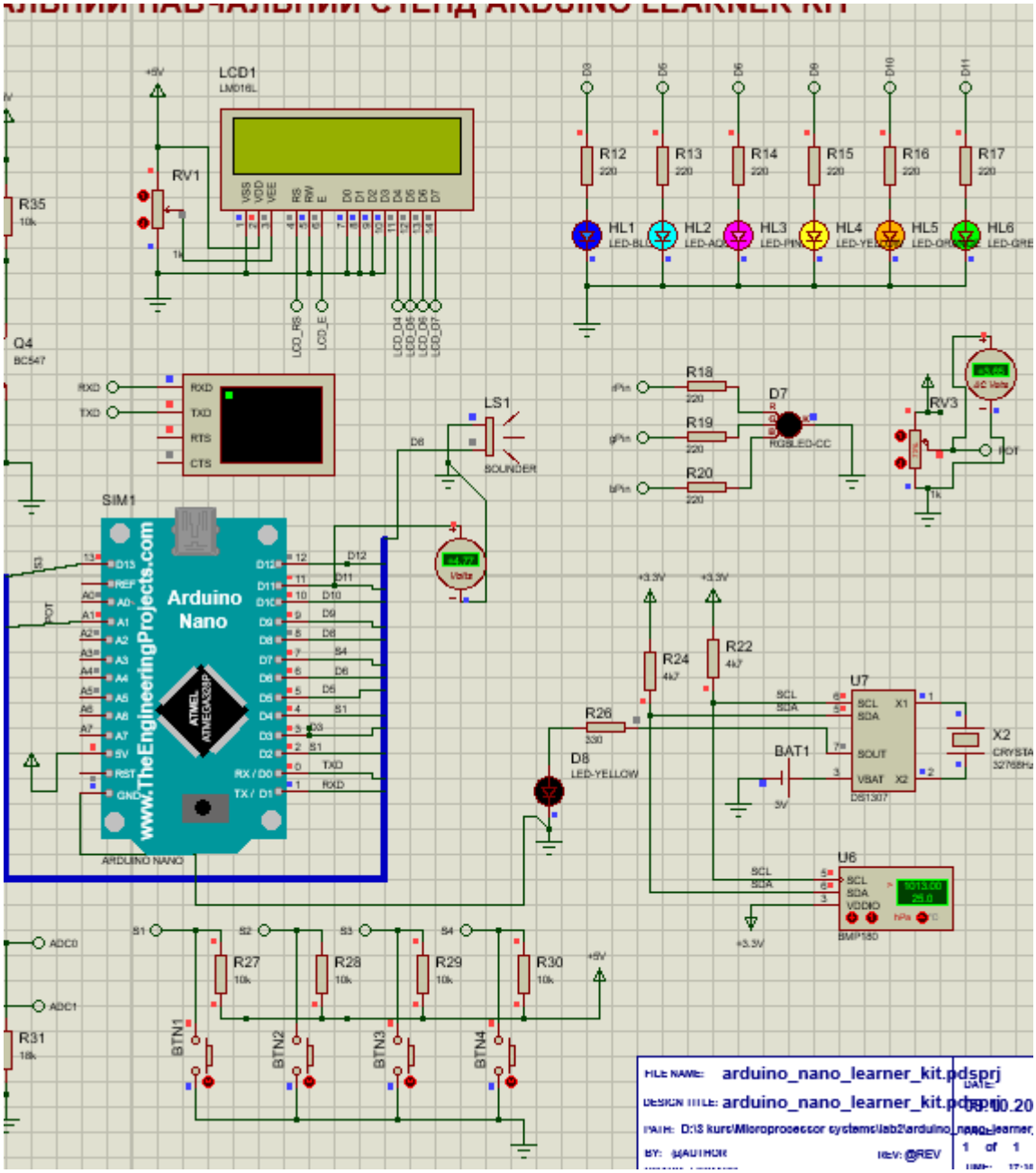

Рис. 11. Виконання завдання

Висновок: У ході виконання лабораторної роботи я дослідив можливості створення мелодії, за допомогою спеціальних бібліотек, в Arduino; дослідив роботу переривань, АЦП та ШІМ в контролерах Arduino.