

4. Аналого-цифрові перетворювачі. Цифро-аналогові перетворювачі

Для передавання аналогового сигналу до МПС використовують аналогово-цифровий перетворювач (АЦП). АЦП сприймає аналоговий сигнал, напругу або струм і перетворює його в цифрове слово, зрозуміле МП. На рис. 4.1 наведена структурна схема модуля (АЦП), що застосовується в МК AVR.

Перетворення «аналог-цифра» здійснюється в 10-бітовому АЦП послідовного наближення. Для його нормальної роботи потрібно три сигнали: вхідний V_{IN} , тактовий $F_{АЦП}$, опорний V_{REF} .

Сигнал V_{IN} поступає від мультиплексора, що комутує вісім аналогових каналів з ліній PA0...PA7 та дві тестових напруги 0 і +1,22В. Вибір джерела сигналу здійснюється програмним способом через регістр ADMUX. Сигнал $F_{АЦП}$ виходить з тактового сигналу F_{CLK} шляхом ділення на коефіцієнт 2...128, що програмно задається регістром ADCSRA. У середині блоку АЦП частота $F_{АЦП}$ ділиться ще раз на 13 або 14 залежно від одноразового або безперервного режиму вимірів. Це і буде істинним значенням частоти дискретизація сигналу F_d .

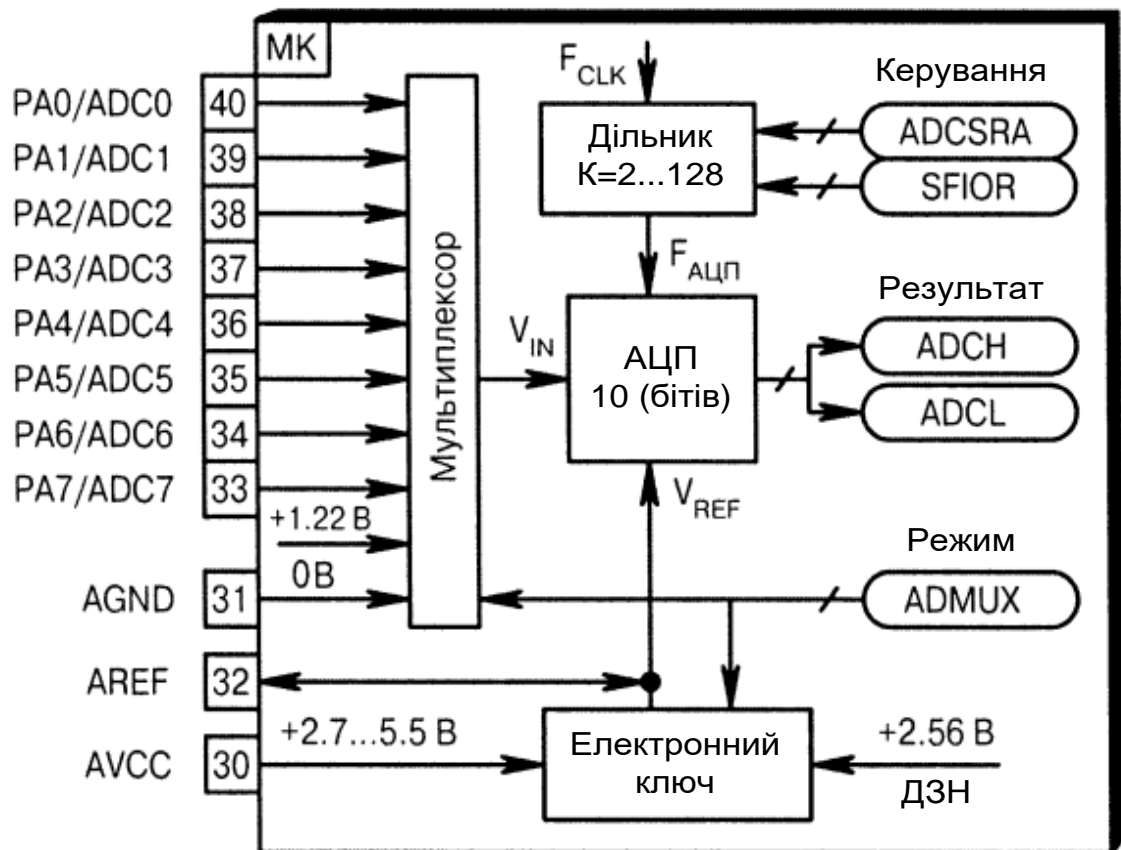


Рисунок 4.1 – Структурна схема модуля АЦП в мікроконтролерах AVR

Припустимо, що $F_{CLK} = 8$ МГц, коефіцієнт ділення встановлений 16, відповідно, $F_{АЦП} = 500$ кГц, $F_d = 38,5$ кГц (13тактів) або $F_d = 35,7$ кГц (14тактів).

Сигнал V_{REF} може поступати з трьох напрямів: від вхідної лінії AREF, від внутрішнього джерела зразкової напруги +2,56 В, від джерела живлення AVCC. Перемикання здійснюється електронним ключем, який керується регістром ADMUX. Вивід AREF має безпосередній електричний зв'язок з модулем АЦП,

тому, для зменшення наведень, його шунтують керамічним конденсатором ємністю 0,1 мкФ.

Результат кожного вимірювання поміщається в регістри ADCH (старші 2 біта) і ADCL (молодші 8 бітів). Разом в двох регістрах утворюється число в діапазоні 0... 1023. Ціна одного ділення – $V_{REF}/1024$. Це справедливо для «чистого» режиму 10 біт. У ATmega328 є ще один режим, що умовно називається 8/10 біт. У ньому вимірювання проводиться з точністю 10 бітів, але відображаються усього лише 8 старших бітів, тобто діапазон складає 0...255, ціна одного ділення $V_{REF}/256$.

При роботі АЦП потрібно на початку програми відключати входні «pull-up» резистори, оскільки лінії ADC0...ADC7 за сумісництвом ще є цифровими входами PA0...PA7.

Аналого-цифровий перетворювач дозволяє зчитати величину напруги на аналогових входах. Це дає можливість зчитувати дані з датчика освітленості, виміряти напругу живлення і т.д. Основні команди для роботи з АЦП в Arduino:

analogReference (type). Параметри:

- type: тип джерела опорної напруги (DEFAULT, INTERNAL, EXTERNAL);
- значення, що повертаються — немає;
- опис: встановлює джерело опорної напруги, що використовується при зчитуванні аналогового сигналу (задає максимальне значення вхідного діапазону). DEFAULT: опорна напруга за замовчуванням, рівна 5В. INTERNAL: внутрішня опорна напруга рівна 1,1В. EXTERNAL: як опорна напруга буде використовуватися напруга, прикладена до виводу AREF (від 0 до 5В).

analogRead (pin). Параметри:

- pin: номер виводу, з якого буде зчитуватися напруга (A0 - A5 для більшості плат, A0 - A7 для Mini та Nano, A0 - A15 для Mega);
- значення, що повертаються: ціле число int (від 0 до 1023);
- опис: зчитує величину напруги з зазначеного аналогового виводу. АЦП перетворювач перетворює вхідну напругу з діапазону 0 - 5В в цілочисельні значення в межах від 0 до 1023 відповідно. Роздільна здатність АЦП становить: 5 В / 1024 значення або 0,0049 В (4.9 мВ) на одне значення. Вхідний діапазон та роздільна здатність можуть змінюватися за допомогою функції *analogReference ()*. Для зчитування значення з аналогового входу потрібно близько 100 мікросекунд, тому максимальна частота опитування виводу приблизно дорівнює 10 000 разів в секунду. Якщо аналоговий вхід ні до чого не підключений, значення, що повертається функцією *analogRead ()*, буде випадковим (змінюється під впливом декількох факторів: величина напруги на інших аналогових входах, наведення від руки поблизу плати).

Розглянемо програму (Prog 1), яка демонструє роботу з АЦП Arduino. Алгоритм програми – значення напруги з потенціометра зчитується внутрішнім АЦП і приймає значення в діапазоні від 0 до 1023. Це значення виводиться в «Монітор Порта» (Serial Monitor) та використовується для формування частоти

мигання світлодіода HL1 та регулювання яскравості світіння світлодіода HL2 (для цього отримане значення ділимо на 4 і отримаємо діапазон регулювання яскравості від 0 до 255).

Лістинг програми Prog 1

```
void setup() {  
    pinMode(11, OUTPUT); //підключення світлодіода HL6  
    pinMode(12, OUTPUT); // підключення світлодіода HL1  
    pinMode(A1, INPUT); // до входу A1 підключаємо потенціометр  
    Serial. begin ( 9600 ) ;  
}  
  
void loop() {  
    int val1 = analogRead(A1);  
    Serial. println ( val1 ) ;  
    int val2 = val1 / 4;  
    analogWrite(11, val2);  
    digitalWrite (12, HIGH);  
    delay(val1) ;  
    digitalWrite (12, LOW);  
    delay(val1);  
}
```

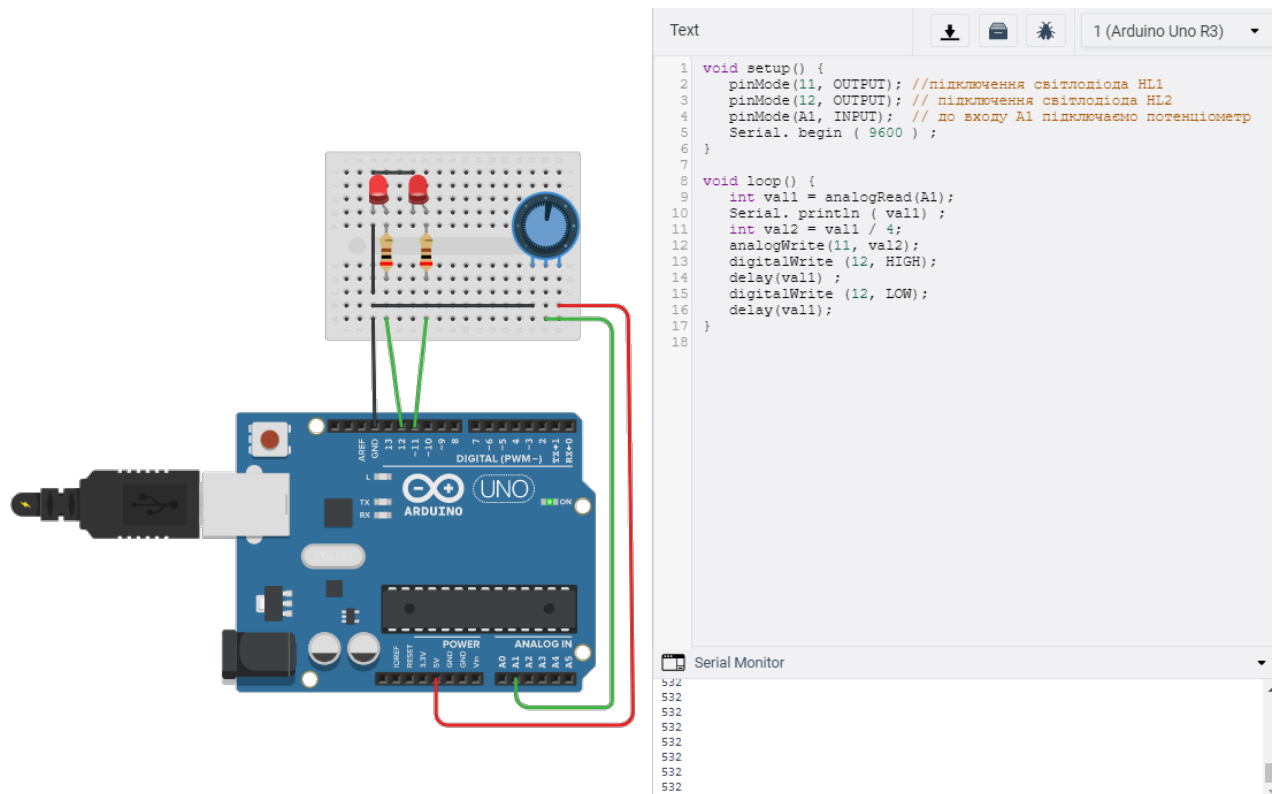


Рисунок 4.2 – Демонстрація роботи програми Prog 1 у середовищі TinkerCad

Цифроаналоговий перетворювач (ЦАП) призначений для перетворення числа, визначеного, як правило, у вигляді двійкових кодів, у напругу або струм пропорційно значенню цифрового коду.

Дуже часто ЦАП входить до складу МПС. У цьому випадку, якщо не потрібна висока швидкодія, цифроаналогове перетворення може бути дуже

просто здійснено за допомогою широтно-імпульсної модуляції (ШІМ). Схема ЦАП з ШІМ наведена на рис.4.3.

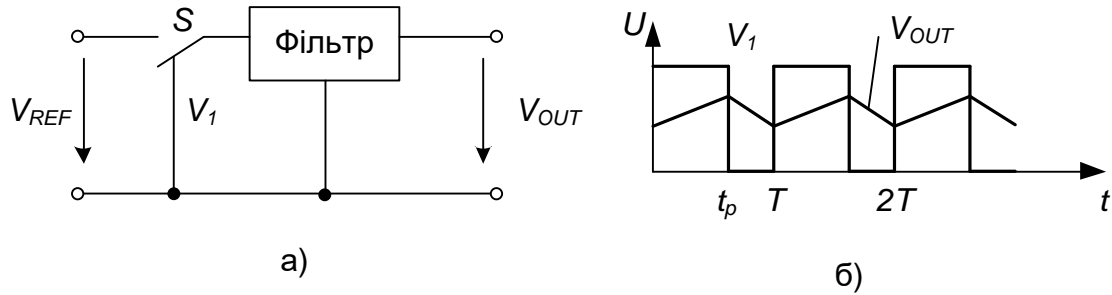


Рисунок 4.3 – ЦАП з широтно-імпульсною модуляцією:
а) структурна схема; б) часова діаграма

Найпростіше організовується цифроаналогове перетворення в тому випадку, якщо МК має вбудовану функцію широко-імпульсного перетворення. Вихід ШІМ керує ключем S . У залежності від заданої розрядності перетворення контролер за допомогою власного таймера/лічильника формує послідовність імпульсів, відносна тривалість яких $\gamma = t_p / T$ визначається співвідношенням:

$$\gamma = \frac{D}{2^N},$$

де N – розрядність перетворення, а D – код перетворення.

Фільтр нижніх частот згладжує імпульси, виділяє середнє значення напруги. У результаті вихідна напруга перетворювача:

$$V_{OUT} = \mathcal{V}_{REF} = \frac{DV_{REF}}{2^N}$$

Розглянута схема забезпечує ідеальну лінійність перетворення, не містить прецизійних елементів (за винятком джерела опорної напруги). Основний її недолік – низька швидкодія.

Для формування сигналів ШІМ Arduino використовує команду ***analogWrite*** (***pin, value***) з параметрами:

- **pin**: вивід, на якому буде формуватися напруга широтно-імпульсної модуляції (ШІМ або PWM);
- **value**: коефіцієнт заповнення — лежить в межах від 0 (завжди вимкнений) до 255 (завжди включений);
- значення, що повертаються: немає;
- опис: формує задану аналогову напругу на виводі у вигляді ШІМ-сигналу. Може використовуватися для зміни яскравості світіння світлодіода або управління швидкістю обертання двигуна. Після виклику *analogWrite()*, на виводі буде безперервно генеруватися ШІМ-сигнал із заданим коефіцієнтом заповнення до наступного виклику функції *analogWrite()* (або до моменту виклику *digitalRead()*, або *digitalWrite()*, взаємодіючих з цим же виводом). Частота ШІМ становить приблизно 490 Гц. На більшості плат Arduino функція *analogWrite* ()

працює з виводами 3, 5, 6, 9, 10 і 11. Функція *analogWrite()* не має нічого спільного з аналоговими виводами і функцією *analogRead()*.

Програма LED4

Програма (Prog2) демонструє зміну яскравості світлодіода методом формування ШІМ сигналу. Алгоритм програми – подаємо на світлодіод 1/3 напруги від 0 до 5В (1,66В) з затримкою 250мс. Напрузі 5В відповідає код 255, напрузі 1,66В відповідає код (85) $255/3$. Далі подаємо напругу 3,33В (код 170), що відповідає 2/3 яскравості світлодіода, та 5В (код 255), що відповідає максимальній яскравості.

Лістинг програми Prog2

```
#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);
}
```

Програма (Prog3) демонструє роботу з RGB світлодіодом. У програмі використовується ШІМ для повільної зміни кольору світіння RGB світлодіоду. Алгоритм програми простий – повільно виключаємо один колір та одночасно повільно включаємо інший колір.

Лістинг програми Prog3

```
const byte rgbPins[3] = {11,10,9};
int dim = 1;

void setup() {
    for(byte i=0; i<3; i++){
        pinMode( rgbPins[i], OUTPUT );
    }
    // початковий стан (Red - on)
    analogWrite(rgbPins[0], 255);
}
```



```

    analogWrite(rgbPins[1], 0);
    analogWrite(rgbPins[2], 0);
}

void loop() {
    for(int i=255; i>=0; i--){
        analogWrite( rgbPins[0], i/dim );//Red-off
        analogWrite( rgbPins[1], (255-i)/dim );//Green-on
        delay(10);
    }

    for(int i=255; i>=0; i--){
        analogWrite( rgbPins[1], i/dim );//Green-off
        analogWrite( rgbPins[2], (255-i)/dim );//Blue-on
        delay(10);
    }

    for(int i=255; i>=0; i--){
        analogWrite( rgbPins[2], i/dim );//Blue-off
        analogWrite( rgbPins[0], (255-i)/dim );//Red-on
        delay(10);
    }
}

```

Для відтворення звукового сигналу, мікроконтролер Arduino, використовує 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 (рис. 4.4).

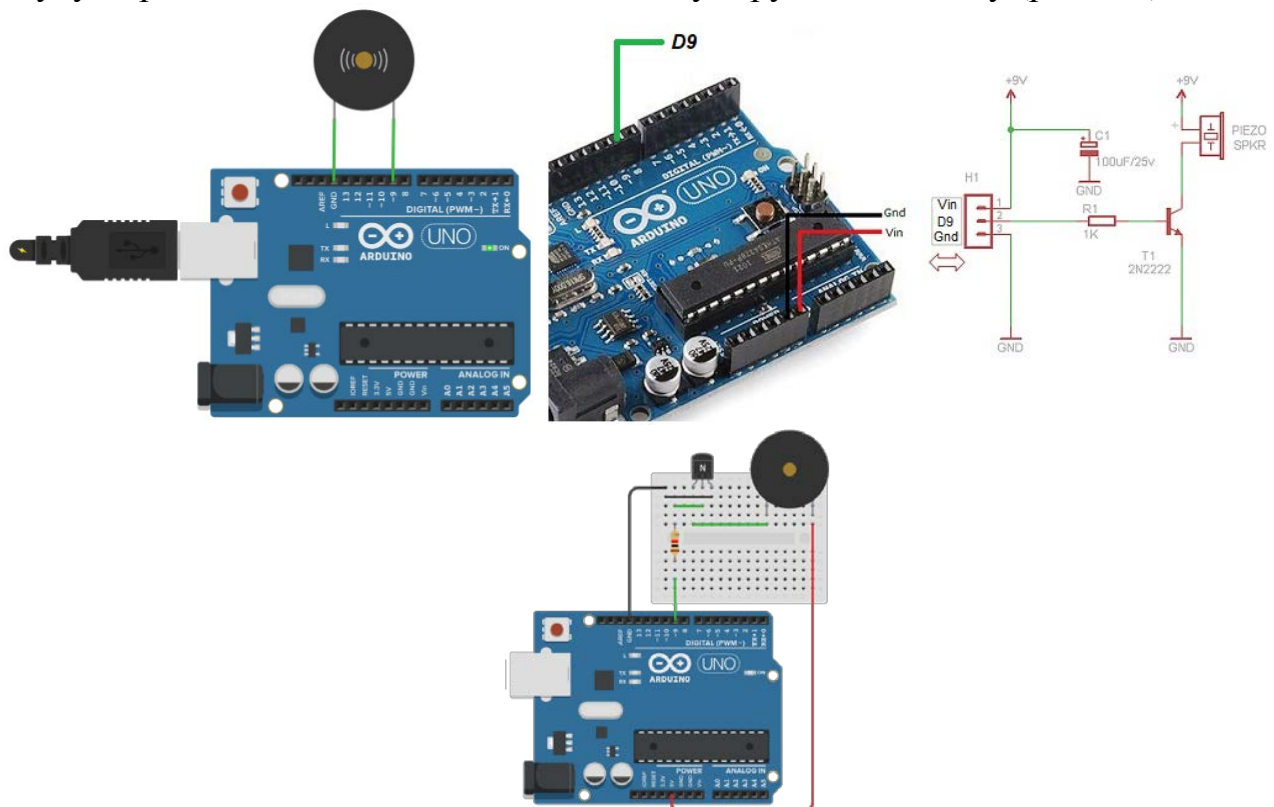


Рисунок 4.4 – Підключення boozер до Arduino безпосередньо та через ключ

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()** 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.

Програма (Prog4) демонструє формування звукового сигналу Arduino. Алгоритм програми простий – аналізується стан тактової кнопки. Якщо вона натиснута, то формується 3-х тональний звуковий сигнал з паузою 0,3 секунди.

Лістинг програми Prog4

```
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);
    }
}
```

Програма (Prog5) демонструє формування музикального фрагменту мікроконтролером Arduino (рис. 4.4). Програму рекомендується перевірити самостійно.

Лістинг програми Prog5

```
void setup() { }
void loop() {
tone(9, 392, 350);
delay(350);
tone(9, 392, 350);
delay(350);
tone(9, 392, 350);
delay(350);
tone(9, 311, 250);
delay(250);
tone(9, 466, 100);
delay(100);
tone(9, 392, 350);
delay(350);
tone(9, 311, 250);
delay(250);
tone(9, 466, 100);
delay(100);
tone(9, 392, 700);
delay(700);
tone(9, 587, 350);
delay(350);
tone(9, 587, 350);
delay(350);

tone(9, 587, 350);
delay(350);
tone(9, 622, 250);
delay(250);
tone(9, 466, 100);
delay(100);
tone(9, 369, 350);
delay(350);
tone(9, 311, 250);
delay(250);
tone(9, 466, 100);
delay(100);
tone(9, 392, 700);
delay(700);
tone(9, 784, 350);
delay(350);
tone(9, 392, 250);
delay(250);

tone(9, 392, 100);
delay(100);
tone(9, 784, 350);
delay(350);
tone(9, 739, 250);
delay(250);
tone(9, 698, 100);
delay(100);
tone(9, 659, 100);
delay(100);
tone(9, 622, 100);
delay(100);
tone(9, 659, 450);
delay(450);

tone(9, 415, 150);
delay(150);
tone(9, 554, 350);
delay(350);
tone(9, 523, 250);
delay(250);
tone(9, 493, 100);
delay(100);
tone(9, 466, 100);
delay(100);
tone(9, 440, 100);
delay(100);
tone(9, 466, 450);
delay(450);

tone(9, 311, 150);
delay(150);
tone(9, 369, 350);
delay(350);
tone(9, 311, 250);
delay(250);
tone(9, 466, 100);
delay(100);
tone(9, 392, 750);
delay(750); delay(5000);
}
```


Д31. Реалізувати програму формування звукового сигналу, тональність якого змінюється у залежності від обертання ручки потенціометра (мінімальна частота 200Гц, максимальна частота 2000Гц). Схему та програму реалізувати у середовищі TinkerCad.

Д32. Реалізувати програму формування звукового сигналу, тональність якого змінюється у залежності від тактової кнопки, що натиснута (кнопок 4) (мінімальна частота 200Гц, максимальна частота 2000Гц). Схему та програму реалізувати у середовищі TinkerCad.

Д33. Реалізувати програму, яка змінює колір RGB світлодіода у залежності від обертання ручки потенціометра. Схему та програму реалізувати у середовищі TinkerCad.