ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра автоматики и телемеханики

Методические указания к лабораторной работе №2

«Работа с пьезодинамиком»

По дисциплине «Информационные технологии»

Цели лабораторной работы:

Изучение основ работы пьезодинамиков и микросхем (сборка схем, подключение, программирование микроконтроллера для корректной работы схем), закрепление знаний о среде Arduino.

МИКРОКОНТРОЛЛЕР

Микроконтроллер — это небольшая микросхема, в которой уже есть процессор, оперативная память и флеш-память. Они бывают разной формы, немного различаются возможностями и производительностью, но суть остаётся неизменной.

Он менее мощный, способен выполнять лишь одну задачу, но зато компактный и дешёвый.

Микроконтроллер часто является мозговым центром платы, отвечающей за определённую задачу: на него приходят все сигналы, поступающие на плату, а он в свою очередь раздаёт команды всем устройствам, подключённым к ней. Микроконтроллеры используются повсеместно: от бытовых кухонных устройств до элементов управления космическим кораблём, от домофонов до систем безопасности в автомобиле, от радиоуправляемых игрушек до роботов на конвейере завода.

Использование в современном микроконтроллере достаточного мощного вычислительного устройства с широкими возможностями, построенного на одной микросхеме вместо целого набора, значительно снижает размеры, энергопотребление и стоимость построенных на его базе устройств.

Arduino uno

Это последняя модель Arduino Rev3 (рисунок 1), оригинальная, произведённая в Италии. Она выполнена на базе процессора ATmega328p с тактовой частотой 16 МГц, обладает памятью 32 кБ и имеет 20 контролируемых контактов ввода и вывода для взаимодействия с внешним миром.

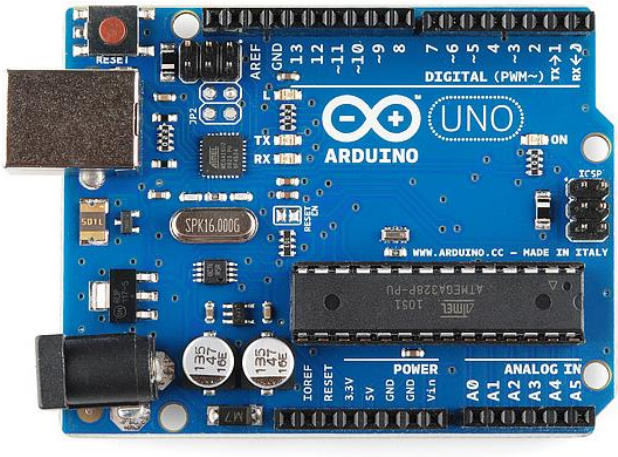

Рисунок 1 - Микроконтроллер Arduino Rev3

Питание

Arduino Uno может питаться как от USB подключения, так и от внешнего источника: батарейки или обычной электрической сети. Источник определяется автоматически.

Платформа может работать при наличии напряжения от 6 до 20 В. Однако при напряжении менее 7 В работа может быть неустойчивой, а напряжение более 12 В может привести к перегреву и повреждению. Поэтому рекомендуемый диапазон: 7−12 В.

На Arduino доступны следующие контакты для доступа к питанию:

- Vin предоставляет тот же вольтаж, что используется для питания платформы. При подключении через USB будет равен 5 В.
- 5V предоставляет 5 В вне зависимости от входного напряжения. На этом напряжении работает процессор. Максимальный допустимый ток, получаемый с этого контакта — 800 мА.
- 3.3V предоставляет 3,3 В. Максимальный допустимый ток, получаемый с этого контакта — 50 мА.

- GND — земля.

Память

Платформа оснащена 32 кБ flash-памяти, 2 кБ из которых отведено под так называемый bootloader. Он позволяет прошивать Arduino с обычного компьютера через USB. Эта память постоянна и не предназначена для изменения по ходу работы устройства. Её предназначение — хранение программы и сопутствующих статичных ресурсов.

Также имеется 2 кБ SRAM-памяти, которые используются для хранения временных данных вроде переменных программы. По сути, это оперативная память платформы. SRAM-память очищается при обесточивании.

Ещё имеется 1 кБ EEPROM-памяти для долговременного хранения данных. По своему назначению это аналог жёсткого диска для Arduino.

Ввод / вывод

На платформе расположены 14 контактов (pins), которые могут быть использованы для цифрового ввода и вывода. Какую роль исполняет каждый контакт, зависит от вашей программы. Все они работают с напряжением 5 В, и рассчитаны на ток до 40 мА. Также каждый контакт имеет встроенный, но отключённый по умолчанию резистор на 20 - 50 кОм. Некоторые контакты обладают дополнительными ролями:

Serial: 0-й и 1-й. Используются для приёма и передачи данных по USB.

Внешнее прерывание: 2-й и 3-й. Эти контакты могут быть настроены так, что они будут провоцировать вызов заданной функции при изменении входного сигнала.

PWM: 3-й, 5-й, 6-й, 9-й, 10-й и 11-й. Могут являться выходами с широтно-импульсной модуляцией (pulse-width modulation) с 256 градациями.

LED: 13-й. К этому контакту подключен встроенный в плату светодиод. Если на контакт выводится 5 В, светодиод зажигается; при нуле — светодиод гаснет.

Помимо контактов цифрового ввода/вывода на Arduino имеется 6 контактов аналогового ввода, каждый из которых предоставляет разрешение в 1024 градации. По умолчанию значение меряется между землёй и 5 В, однако

возможно изменить верхнюю границу, подав напряжение требуемой величины на специальный контакт AREF.

Кроме этого на плате имеется входной контакт Reset. Его установка в логический ноль приводит к сбросу процессора. Это аналог кнопки Reset обычного компьютера.

ПЬЕЗОДИНАМИК

Пьезоизлучатель звука (рисунок 2) (англ. buzzer) переводит переменное напряжение в колебание мембраны, которая в свою очередь создаёт звуковую волну.

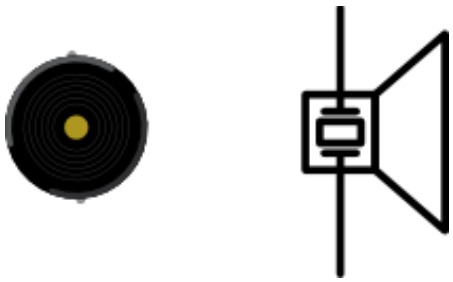

Рисунок 2 - Пьезодинамик

Иначе говоря, пьезодинамик — это конденсатор, который звучит при зарядке и разрядке.

Основные характеристики

Таблица 1 - Основные характеристики

Рекомендуемое (номинальное) напряжение	V	Вольт
Громкость (на заданном расстоянии)	P	Децибелл
Пиковая частота	f_P	Герц
Ёмкость	C	Фарад

Амплитудно-частотная характеристика (АЧХ) определяет громкость звука в зависимости от частоты управляющего сигнала, который и определяет высоту звучащей ноты.

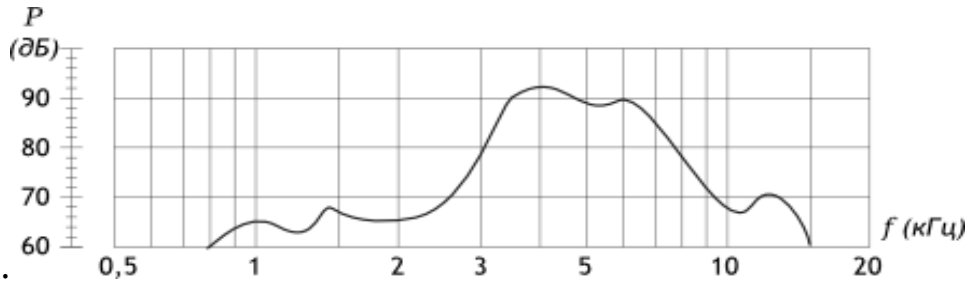

Рисунок 3 - Зависимость громкости звука от частоты

Идеальная АЧХ — это прямая, т.е. одинаковая громкость вне зависимости от частоты. Но мир не идеален и разные виды излучателей имеют разные отклонения от идеала.

Подключение напрямую

Пьезодинамик потребляет всего пару мА, поэтому можно смело подключать его прямо к микроконтроллеру (рисунок 4).

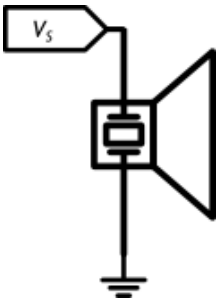

Рисунок 4 - прямое подключение к микроконтроллеру

Для звучания нужно подавать на динамик квадратную волну (рисунок 5). Какой частоты будет волна, такой частоты будет и звук.

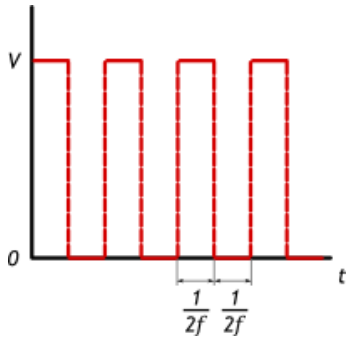

Рисунок 5 - квадратная волна, подаваемая на динамик

КОНДЕНСАТОР

Конденсатор — крошечный аккумулятор, который очень быстро заряжается и очень быстро разряжается. Виды конденсаторов, используемых в Arduino, показаны на рисунке 6.

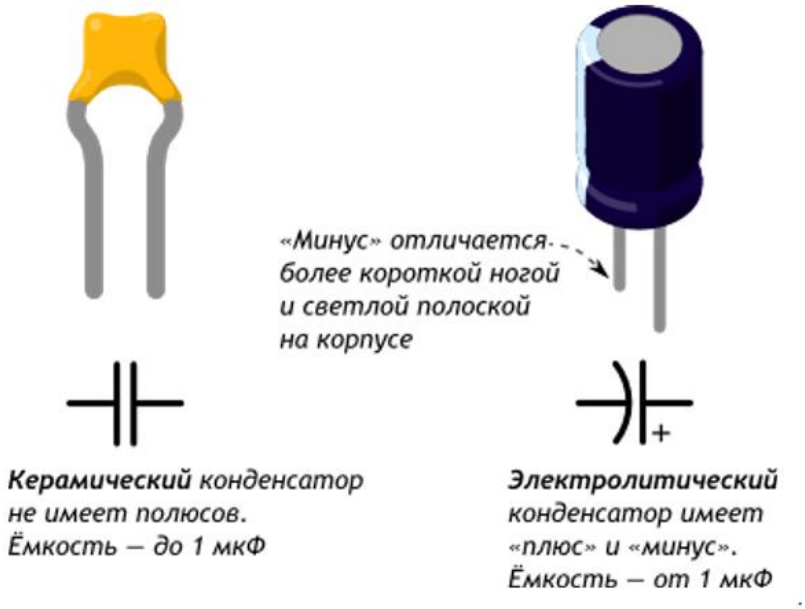

Рисунок 6 - Виды конденсаторов, используемых в Arduino

Основные характеристики

Таблица 2 - основные характеристики

Ёмкость (номинал)	C	Фарад
Точность (допуск)	±	%
Максимальное напряжение	V	Вольт

Кодирование номинала

Рисунок 7 - Пример кодировки номинала конденсатора

Номинал в пФ записан на корпусе (рисунок 7). Первые 2 цифры — основание, 3-я — множитель. Например:

- 220 = 22 × 100 пФ = 22 пФ
- 471 = 47 × 101 пФ = 470 пФ
- 103 = 10 × 103 пФ = 10 000 пФ = 10 нФ
- 104 = 10 × 104 пФ = 100 000 пФ = 100 нФ

Поведение

Если подаваемое напряжение больше внутреннего накопленного, конденсатор будет заряжаться. Если внешнее напряжение меньше внутреннего, конденсатор будет отдавать заряд. Разрядка и зарядка конденсатора изображены на рисунке 8.

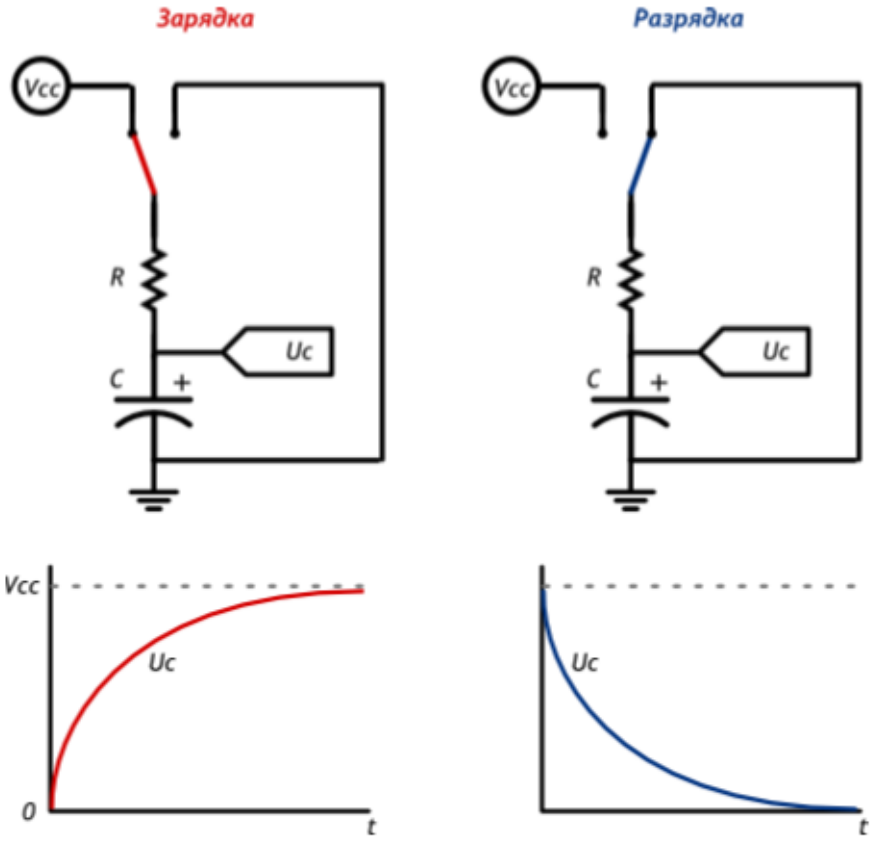

Рисунок 8 – Зарядка и разрядка конденсатора

Время заряда и разряда

Для связывания уровня заряда конденсатора с временем используют понятие «постоянной времени τ»:

$$\tau = R \times C$$

-□ За τ секунд конденсатор заряжается или разряжается на 63% □

- За 5×τ секунд конденсатор заряжается или разряжается на 99% □

- Если резистора в схеме нет, его роль исполняет паразитное сопротивление проводов, разъѐмов, дорожек, составляющее доли Ома

ИНВЕРТИРУЮЩИЙ ТРИГГЕР ШМИДТА

Триггер Шмидта — это устройство, которое преобразовывает, вероятно, нестабильный аналоговый сигнал в стабильный цифровой (рисунок 9). Суть такова:

- Входящий аналоговый сигнал может быть либо восходящим (↗), либо нисходящим (↘)
- В триггере определены 2 пороговых значения: ≈1,6 В — для восходящего сигнала и ≈0,9 В — для нисходящего
- Выход триггера становится логической единицей (5 В), только когда восходящий сигнал проходит свой порог в 1,6 В; при этом прохождение восходящим сигналом нижнего порога в 0,9 В будет проигнорированно
- Аналогично, выход триггера становится логическим нулём (0 В), только когда нисходящий сигнал проходит свой порог в 0,9 В; при этом прохождение нисходящим сигналом нижнего порога в 1,6 В будет проигнорированно
- В остальное время значение выхода сохраняется на прежнем уровне

Зона неопределённости между порогами называется гистерезисом.

Рисунок 9 - Микросхема 74HC17 (Инвертирующий триггер Шмидта)

В данном чипе собраны инвертирующие триггеры. Это означает, что в дополнение к сказанному, выходной цифровой сигнал инвертируется: ноль становится единицей, единица — нулём.

Триггер Шмитта может быть полезен для аппаратной стабилизации сигналов. Таких, как сигнал от тактовой кнопки, подверженной эффекту дребезга.

Ход работы

1 Эксперимент «Азбука Морзе»:

1.1 Собираем схему, изображенную на рисунках 10 и 11;

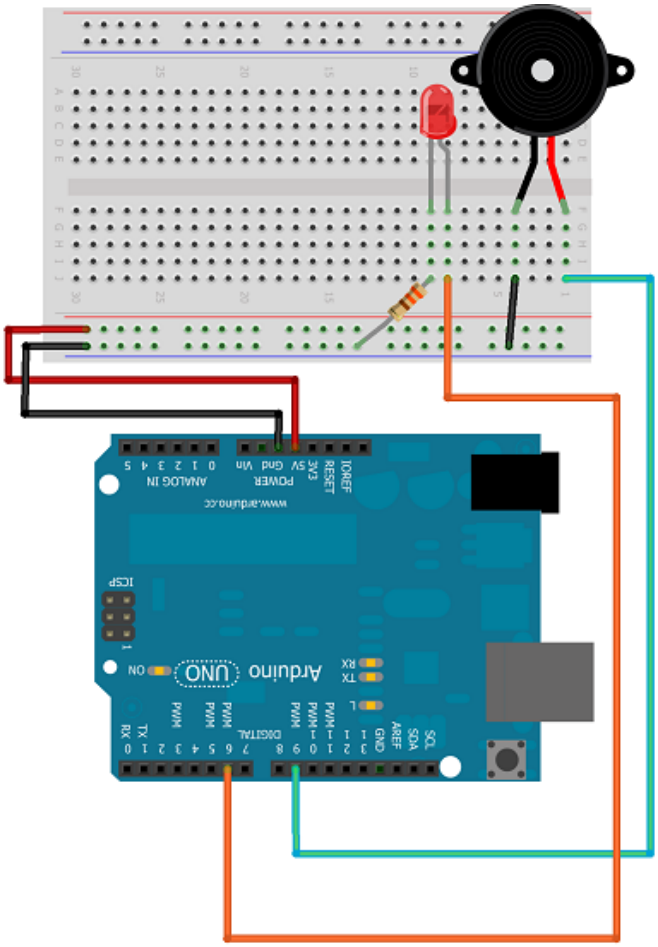

Рисунок 10 - Визуальная схема сборки

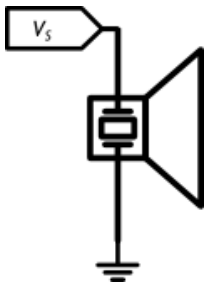

Рисунок 11 - Принципиальная схема сборки

1.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
int soundPin = 13;
int dotDelay = 50;

char theword[] = "help";
```

```
long time;
int rate = 100;

void sound(int duration)
{
     time = millis();
     while (millis() - time < duration) {
         digitalWrite(soundPin, HIGH);
         delayMicroseconds(rate);
         digitalWrite(soundPin, LOW);
         delayMicroseconds(rate);
     }
}

void dot()
{
    sound(dotDelay);
    delay(dotDelay);
}

void dash()
{
    sound(3 * dotDelay);
    delay(dotDelay);
}

void letterEnd()
{
    delay(2 * dotDelay);
}

void wordEnd()
{
    delay(6 * dotDelay);
}

void morseWord(char theword[])
{
    int len = strlen(theword);
    for(int i  = 0; i < len; ++i)
        morseLetter(theword[i]);
    wordEnd();
}

void morseLetter(char c)
{
    switch(c) {
        case 'a': dot();dash(); break;
        case 'b': dash();dot();dot();dot(); break;
        case 'c': dash();dot();dash();dot(); break;
        case 'd': dash();dot();dot(); break;
        case 'e': dot(); break;
        case 'f': dot();dot();dash();dot(); break;
        case 'g': dash();dash();dot(); break;
        case 'h': dot();dot();dot(); break;
        case 'i': dot();dot(); break;
        case 'j': dot();dash();dash();dash(); break;
        case 'k': dash();dot();dash(); break;
        case 'l': dot();dash();dot();dot(); break;
        case 'm': dash();dash(); break;
        case 'n': dash();dot(); break;
```

```
            case 'o': dash();dash();dash(); break;
            case 'p': dot();dash();dash();dot(); break;
            case 'q': dash();dash();dot();dash(); break;
            case 'r': dot();dash();dot(); break;
            case 's': dot();dot();dot(); break;
            case 't': dash(); break;
            case 'u': dot();dot();dash(); break;
            case 'v': dot();dot();dot();dash(); break;
            case 'w': dot();dash();dash(); break;
            case 'x': dash();dot();dot();dash(); break;
            case 'y': dash();dot();dash();dash(); break;
            case 'z': dash();dash();dot();dot(); break;
            case ' ': wordEnd(); break;
        }
        letterEnd();
}

void setup()
{
    pinMode(soundPin, OUTPUT);
}

void loop()
{
    morseWord(theword);
}
```

1.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

2 Эксперимент «Азбука Морзе со светодиодным индикатором»:

2.1 Собираем схему, изображенною на рисунках 12 и 13;

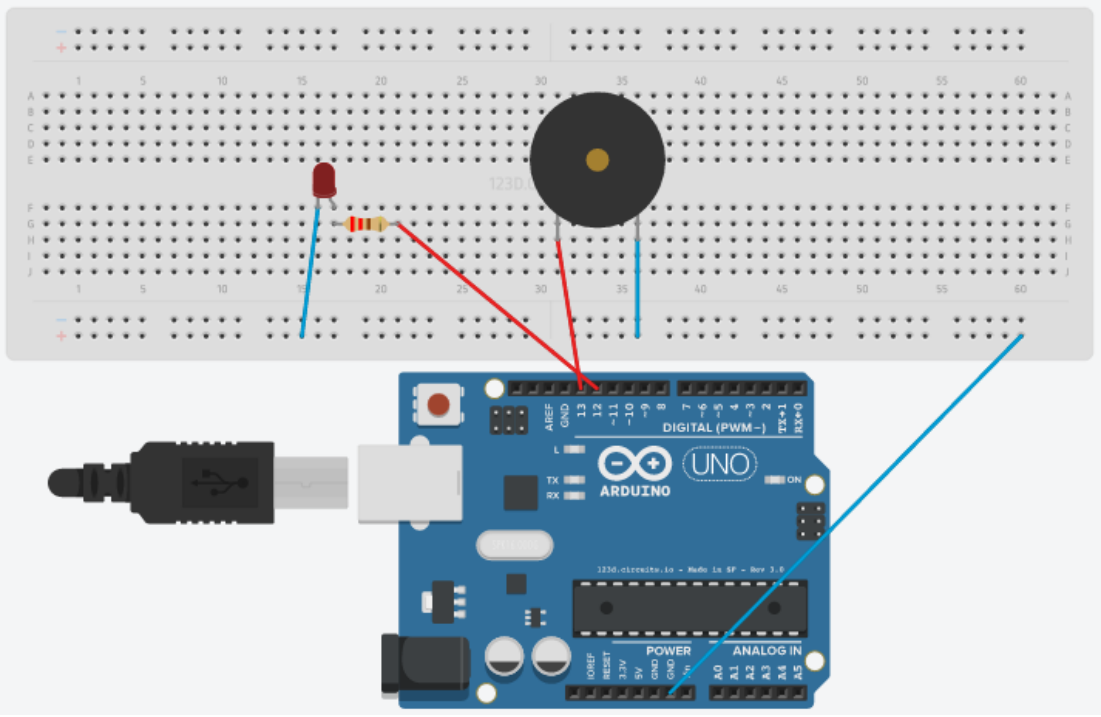

Рисунок 12 - Визуальная схема сборки

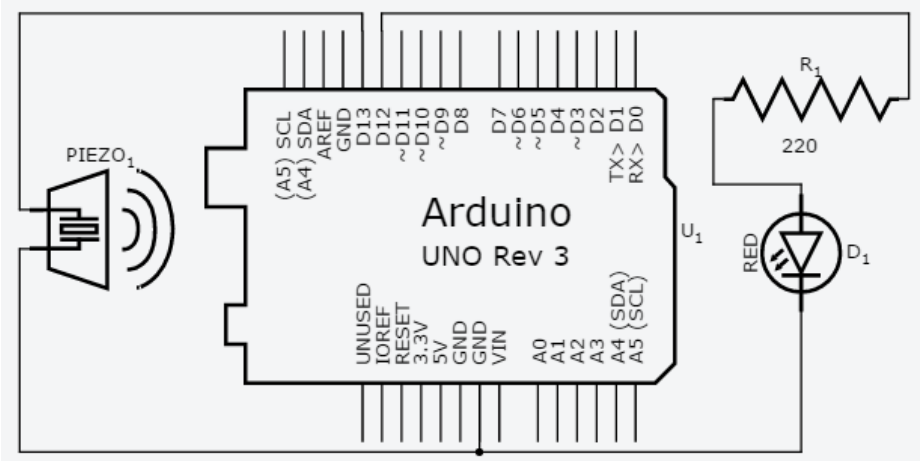

Рисунок 13 - Принципиальная схема сборки

2.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
int LedPin = 12;
int soundPin = 13;
int dotDelay = 50;

char theword[] = "help";

long time;
int rate = 100;

void sound(int duration)
{
     time = millis();
     while (millis() - time < duration) {
         digitalWrite(soundPin, HIGH);
         delayMicroseconds(rate);
         digitalWrite(soundPin, LOW);
         delayMicroseconds(rate);
     }
}

void dot()
{
    sound(dotDelay);
    digitalWrite(LedPin, HIGH);
    delay(dotDelay);
    digitalWrite(LedPin, LOW);
}

void dash()
{
    sound(3 * dotDelay);
    digitalWrite(LedPin, HIGH);
    delay(3 * dotDelay);
    digitalWrite(LedPin, LOW);
}

void letterEnd()
{
    delay(2 * dotDelay);
```

```
}


void wordEnd()
{
    delay(6 * dotDelay);
}

void morseWord(char theword[])
{
    int len = strlen(theword);
    for(int i  = 0; i < len; ++i)
        morseLetter(theword[i]);
    wordEnd();
}

void morseLetter(char c)
{
    switch(c) {
        case 'a': dot();dash(); break;
        case 'b': dash();dot();dot();dot(); break;
        case 'c': dash();dot();dash();dot(); break;
        case 'd': dash();dot();dot(); break;
        case 'e': dot(); break;
        case 'f': dot();dot();dash();dot(); break;
        case 'g': dash();dash();dot(); break;
        case 'h': dot();dot();dot(); break;
        case 'i': dot();dot(); break;
        case 'j': dot();dash();dash();dash(); break;
        case 'k': dash();dot();dash(); break;
        case 'l': dot();dash();dot();dot(); break;
        case 'm': dash();dash(); break;
        case 'n': dash();dot(); break;
        case 'o': dash();dash();dash(); break;
        case 'p': dot();dash();dash();dot(); break;
        case 'q': dash();dash();dot();dash(); break;
        case 'r': dot();dash();dot(); break;
        case 's': dot();dot();dot(); break;
        case 't': dash(); break;
        case 'u': dot();dot();dash(); break;
        case 'v': dot();dot();dot();dash(); break;
        case 'w': dot();dash();dash(); break;
        case 'x': dash();dot();dot();dash(); break;
        case 'y': dash();dot();dash();dash(); break;
        case 'z': dash();dash();dot();dot(); break;
        case ' ': wordEnd(); break;
    }

    letterEnd();
}

void setup()
{
    pinMode(soundPin, OUTPUT);
}

void loop()
{
    morseWord(theword);
}
```

2.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

3 Эксперимент «Терменвокс»:

3.1 Собираем схему, изображенную на рисунках 14 и 15;

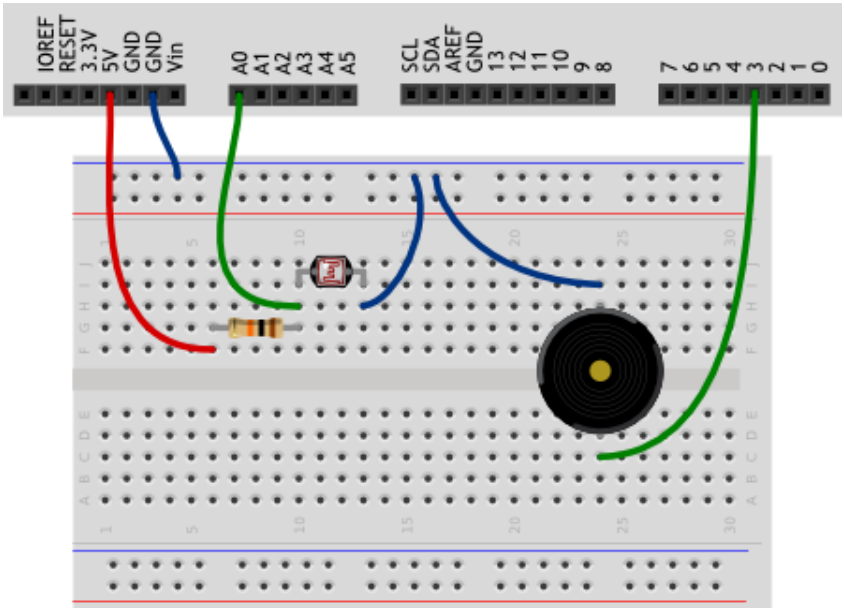

Рисунок 14 - Визуальная схема сборки

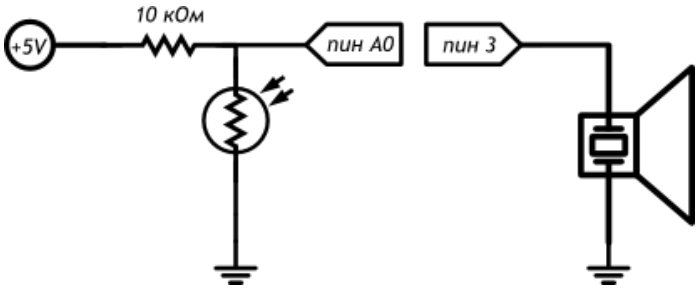

Рисунок 15 - Принципиальная схема сборки

3.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
#define BUZZER_PIN  3
#define LDR_PIN     A0

void setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop()
{
  int val, frequency;
  val = analogRead(LDR_PIN);
  frequency = map(val, 0, 1023, 3500, 4500);
```

```
  tone(BUZZER_PIN, frequency, 20);
}
```

3.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

4 Эксперимент «Мерзкое пианино»:

4.1 Собираем схему, изображенную на рисунках 16 и 17;

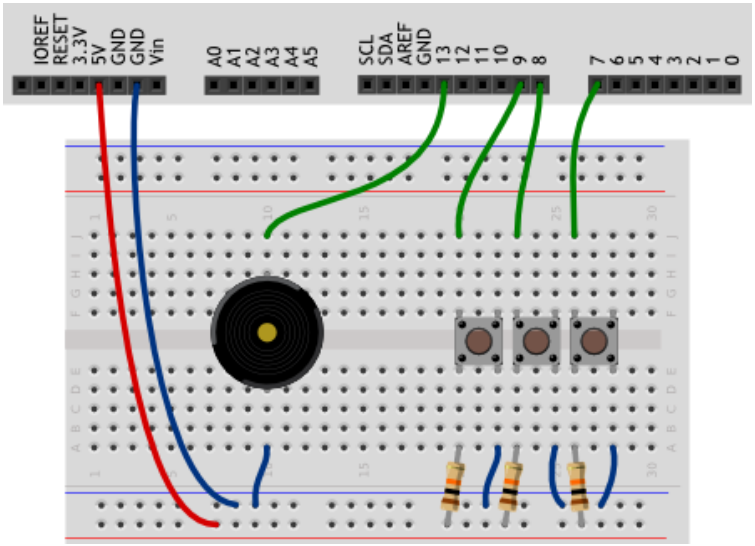

Рисунок 16 - Визуальная схема сборки

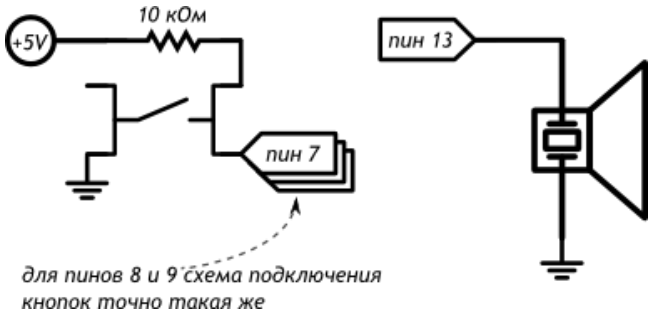

Рисунок 17 - Принципиальная схема сборки

4.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
#define BUZZER_PIN    13
#define FIRST_KEY_PIN 7
#define KEY_COUNT     3


void setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop()
{
```

```
  for (int i = 0; i < KEY_COUNT; ++i) {
    int keyPin = i + FIRST_KEY_PIN;
    boolean keyUp = digitalRead(keyPin);

    if (!keyUp) {
      int frequency = 3500 + i * 500;
      tone(BUZZER_PIN, frequency, 20);
    }
  }
}
```

4.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

5 Эксперимент «Мерзкое пианино с 5 кнопками»:

5.1 Собираем схему, изображенную на рисунках 18 и 19;

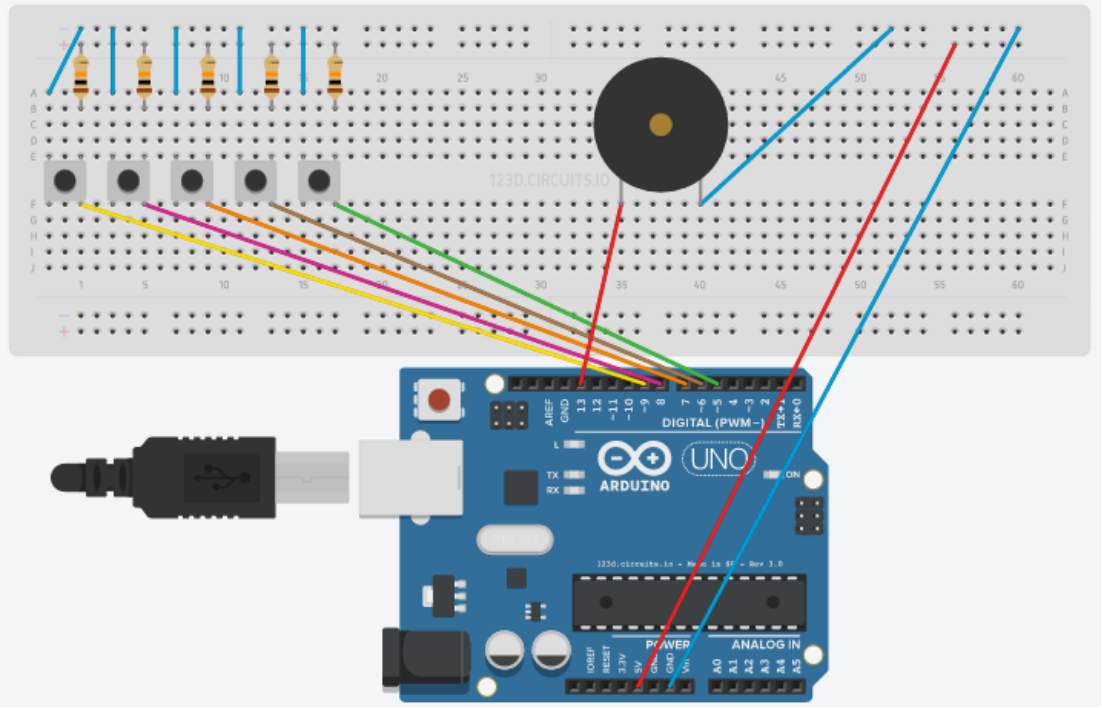

Рисунок 18 - Визуальная схема сборки

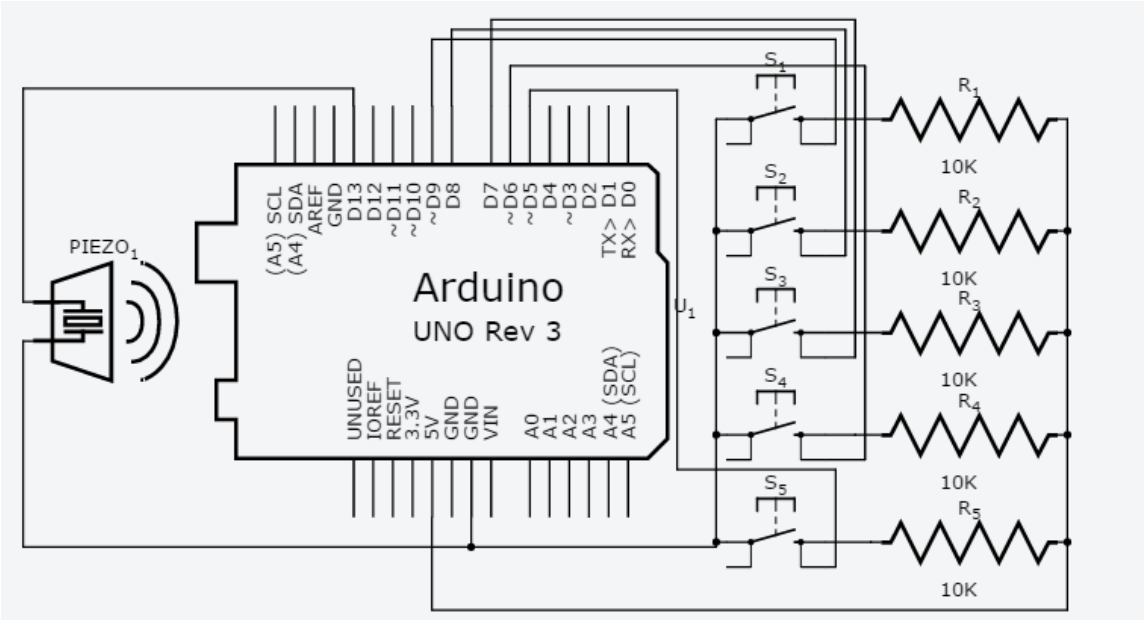

Рисунок 19 - Принципиальная схема сборки

5.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
#define BUZZER_PIN    13
#define FIRST_KEY_PIN 5
#define KEY_COUNT     5
int music[] = {2500, 3000, 3500, 4000, 4500};
void setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop()
{
  for (int i = 0; i < KEY_COUNT; ++i) {
    int keyPin = i + FIRST_KEY_PIN;
    boolean keyUp = digitalRead(keyPin);

    if (!keyUp) {
      tone(BUZZER_PIN, music[keyPin-5], 20);
    }
  }
}
```

5.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

6 Эксперимент «Кнопочные ковбои»:

6.1 Собираем схему, изображённую на рисунках 20 и 21;

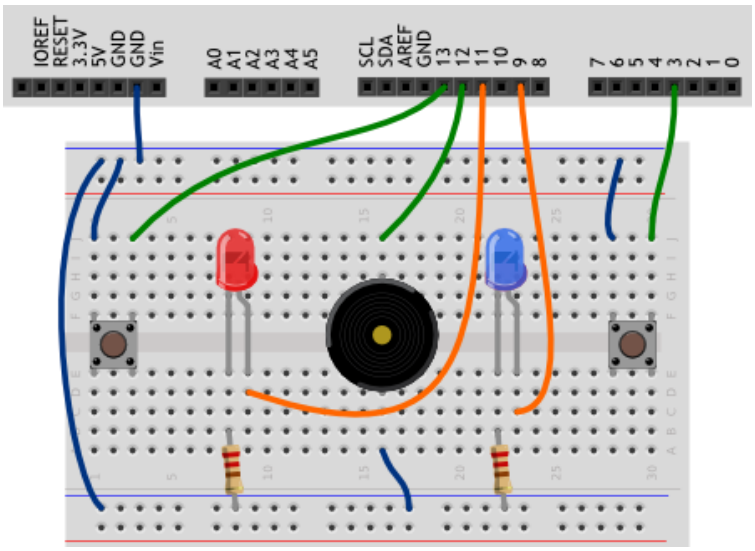

Рисунок 20 - Визуальная схема сборки

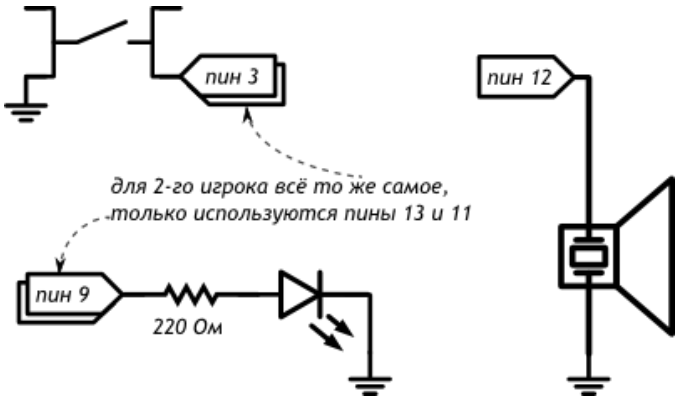

Рисунок 21 - Принципиальная схема сборки

6.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
#define BUZZER_PIN   12
#define PLAYER_COUNT 2
int buttonPins[PLAYER_COUNT] = {3, 13};
int ledPins[PLAYER_COUNT] = {9, 11};

void setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
  for (int player = 0; player < PLAYER_COUNT; ++player) {
    pinMode(ledPins[player], OUTPUT);
    pinMode(buttonPins[player], INPUT_PULLUP);
  }
}

void loop()
{
  delay(random(2000, 7000));
  tone(BUZZER_PIN, 3000, 250);

  for (int player = 0; ; player = (player+1) % PLAYER_COUNT) {
```

```
      if (!digitalRead(buttonPins[player])) {
        digitalWrite(ledPins[player], HIGH);
        tone(BUZZER_PIN, 4000, 1000);
        delay(1000);
        digitalWrite(ledPins[player], LOW);
        break;
      }
    }
  }
```

6.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

7 Эксперимент «Кнопочные ковбои с полосой здоровья»:

7.1 Собираем схему, изображенную на рисунке 22;

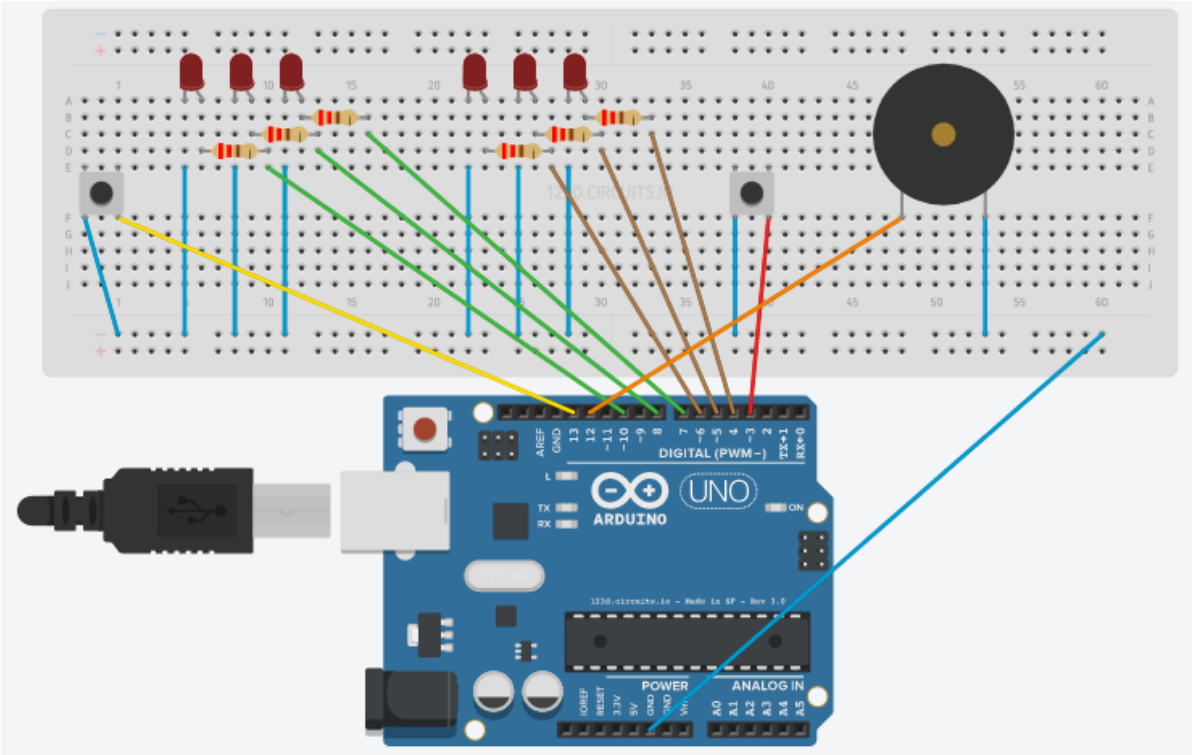

Рисунок 22 - Визуальная схема сборки

7.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
#define BUZZER_PIN   12   // пин с пищалкой
#define PLAYER_COUNT 2   // количество игроков-ковбоев
#define LED_COUNT 3 // кол-во светодиовов для одного игрока
int buttonPins[PLAYER_COUNT] = {3, 13};
int players_led[PLAYER_COUNT][LED_COUNT] = {{4,5,6},{7,8,10}};
int players_health[PLAYER_COUNT] = {2,2}; // Жизнь в диапазон от -1 до 2
int count_kill = 0; // Кол-во убитых игроков

// Отбираем 1 жизнь у игрока с индексом player
void HealthMinus(int player){
  digitalWrite(players_led[player][players_health[player]], LOW);
  players_health[player]--;
}
```

```
// Звук победы
void SoundWin(){
    for(int i = 0; i <= 2; i++){
        tone(BUZZER_PIN, 4000, 200);
        delay(400);
    }
}

// Возвращаем все в исходное состояние
void Clear(){
  count_kill = 0;
  for(int i =0; i < PLAYER_COUNT; ++i){
    for(int j =0; j < LED_COUNT; ++j){
      digitalWrite(players_led[i][j], HIGH);
    }
    players_health[i] = 2;
  }
}

void setup()
{
  pinMode(BUZZER_PIN, OUTPUT);
  for (int player = 0; player < PLAYER_COUNT; ++player) {
    for(int i = 0; i < LED_COUNT; ++i){
      pinMode(players_led[player][i], OUTPUT);
      digitalWrite(players_led[player][i], HIGH);
    }
    pinMode(buttonPins[player], INPUT_PULLUP);
  }
}

void loop()
{
  while (true){
  // Даём сигнал «пли!», выждав случайное время от 2 до 7 сек
  delay(random(2000, 7000));
  tone(BUZZER_PIN, 3000, 250);
  for (int player = 0; ; player = (player+1) % PLAYER_COUNT) {
    // Запрещаем убитым игрокам участвовать в новом раунде
    if(players_health[player] != -1){
    // Если игрок номер «player» нажал кнопку...
    if (!digitalRead(buttonPins[player])) {
      // Отнимаем жизни у всех проигравших игроков
      for(int i =0; i < PLAYER_COUNT; ++i){
        if(i != player){
          HealthMinus(i);
          // Проверяем, умер ли проигравший игрок
          if (players_health[i] == -1){
            count_kill++;
          }
        }
      }
      tone(BUZZER_PIN, 4000, 1000);
      delay(1000);
      break;
    }
  }
  }
  //Если в живых остался лишь один игрок
  if(count_kill == PLAYER_COUNT-1){
```

```
      SoundWin();
      break;
    }
  }
    Clear();
    delay(2000);
}
```

7.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

8 Эксперимент «Перетягивание каната»:

8.1 Собираем схему, изображенную на рисунках 23 и 24;

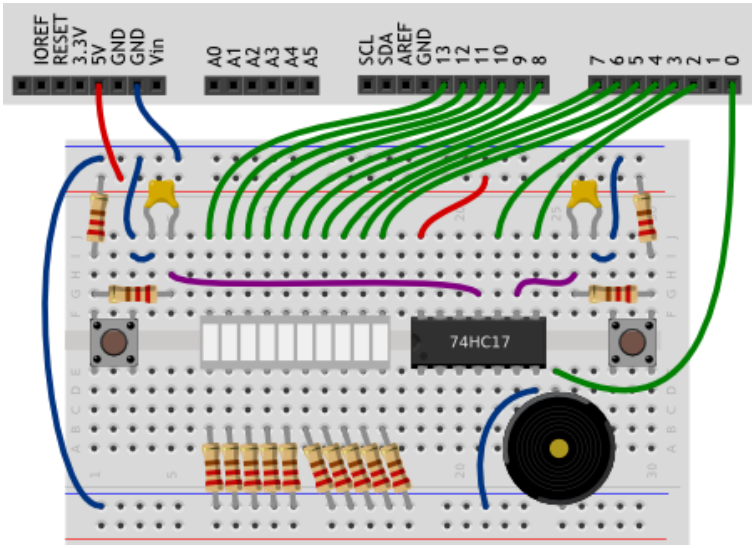

Рисунок 23 - Визуальная схема сборки

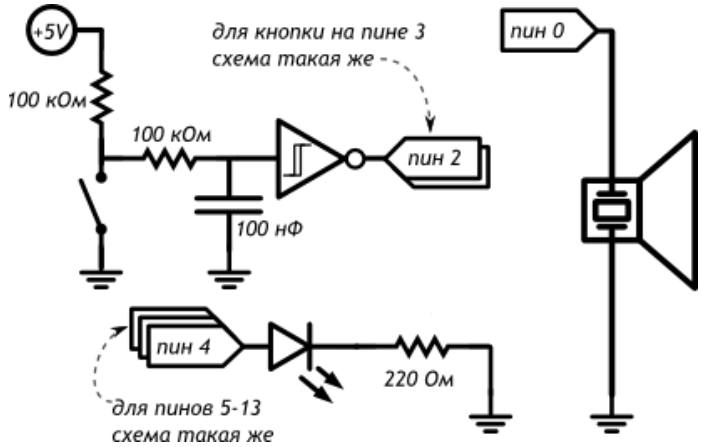

Рисунок 24 - Принципиальная схема сборки

8.2 Для приведения данной схемы в рабочее состояние напишем программный код, представленный ниже;

```
#define BUZZER_PIN     0
#define FIRST_BAR_PIN  4
#define BAR_COUNT      10
#define MAX_SCORE      20
```

```
volatile int score = 0;

void setup()
{
  for (int i = 0; i < BAR_COUNT; ++i)
    pinMode(i + FIRST_BAR_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  attachInterrupt(INT1, pushP1, FALLING);
  attachInterrupt(INT0, pushP2, FALLING);
}

void pushP1() { ++score; }
void pushP2() { --score; }
void loop()
{
  tone(BUZZER_PIN, 2000, 1000);
  while (abs(score) < MAX_SCORE) {
    int bound = map(score, -MAX_SCORE, MAX_SCORE, 0, BAR_COUNT);
    int left = min(bound, BAR_COUNT / 2 - 1);
    int right = max(bound, BAR_COUNT / 2);
    for (int i = 0; i < BAR_COUNT; ++i)
      digitalWrite(i + FIRST_BAR_PIN, i >= left && i <= right);
  }
  tone(BUZZER_PIN, 4000, 1000);
  while (true) {}
}
```

8.3 Проверяем работоспособность собранной схемы, подключив ее к источнику тока.

Вопросы для самопроверки:

- Что такое пьезоэлемент?
- Можно ли устроить полифоническое звучание с помощью функции tone?
- Почему разные «ноты», издаваемые пищалкой, звучат с разной громкостью?
- В чем разница между INPUT и INPUT PULLUP?
- Можно ли поместить в один массив элементы типа boolean и int?
- Для чего испльзуется инвертирующий триггер Шмидта?

ОГЛАВЛЕНИЕ