Практическое занятие №
Тема: «Пианино»

Цель работы: приобрести практические навыки по подключению и программированию кнопок и зумеров на платформе Arduino.

Последовательность выполнения работы:

– Собрать схемы на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно заданию в примере.

Содержание отчета:

– название практического занятия, его цель;

– фото или скриншоты собранной схемы;

– написанный программный код вставить текстом, Courier New, 12 кегль, одинарный отступ без абзацев;

– вывод о проделанной работе;

– файл Fritzing с принципиальной и монтажной схемой.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

ПОДКЛЮЧЕНИЕ КНОПОК

Кнопка (тактовый переключатель) - это простейшее электромеханическое устройство, которое замыкает или размыкает электрическую цепь при нажатии.

Рисунок 1 – Кнопка

1.1 Подключение с внешним подтягивающим резистором

Кнопка → GND

├── Резистор 10кОм → VCC (+5V)

└── Пин Arduino

При отпущенной кнопке: пин через резистор подключен к VCC → HIGH

При нажатой кнопке: пин подключен к GND → LOW

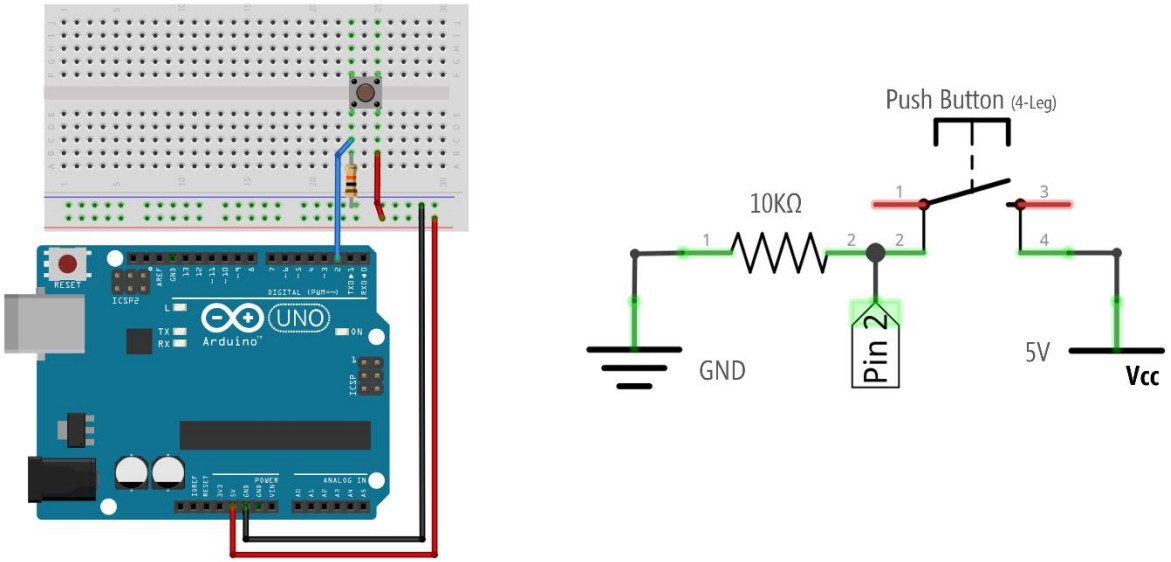

Рисунок 2 – Подключение с внешним подтягивающим резистором

1.2 Подключение с INPUT_PULLUP (внутренний резистор)

pinMode(pin, INPUT_PULLUP);

Arduino имеет встроенные подтягивающие резисторы (~20кОм), которые можно активировать программно.

Особенности логики:

- Отпущенная кнопка: digitalRead() == HIGH
- Нажатая кнопка: digitalRead() == LOW

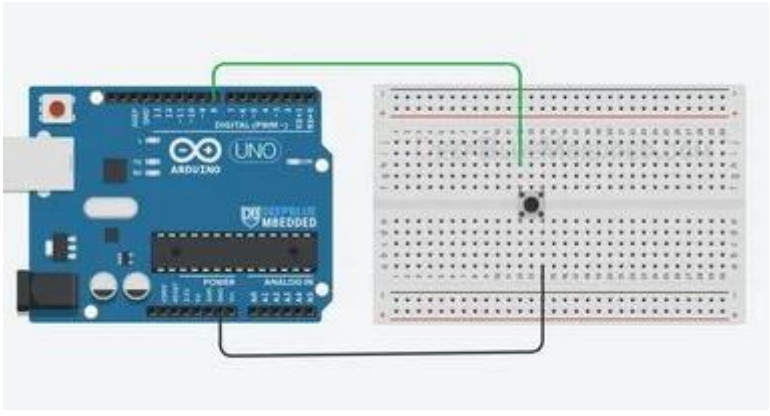

Рисунок 3 – Подключение с внутренним резистором

Дребезг контактов

При нажатии кнопки возникают множественные быстрые замыкания/размыкания (10-50мс). Необходима антидребезговая обработка:

Способы борьбы с дребезгом:

- Программная задержка (20-50мс)
- Аппаратный RC-фильтр
- Использование триггера Шмитта

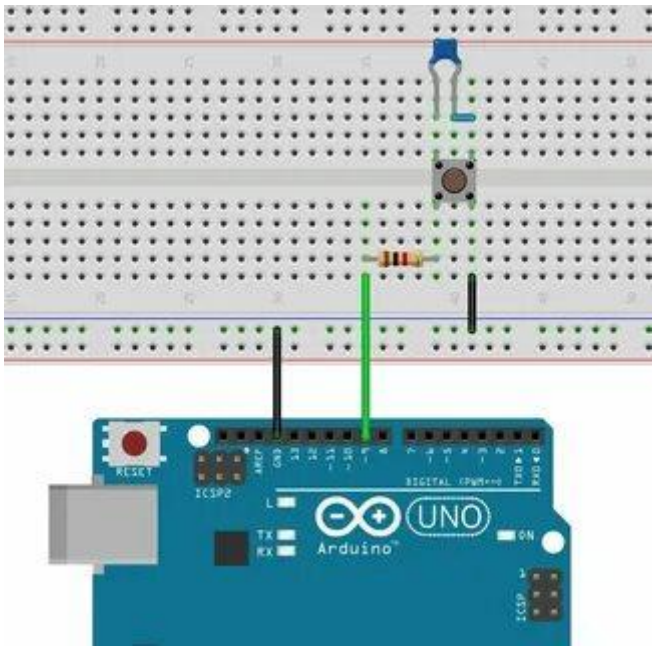

Рисунок 4 – Аппаратный RC-фильтр

Пример обработки кнопки с антидребезгом

```
bool debounceRead(int pin) {
  if (digitalRead(pin) == LOW) {
// Если кнопка нажата
    delay(50);
// Ждем 50мс
    if (digitalRead(pin) == LOW) {
// Проверяем еще раз
      return true;
// Кнопка действительно нажата
    }
  }
  return false;
}
```

ЗУММЕР И ЕГО ПОДКЛЮЧЕНИЕ

Типы зуммеров

Пассивный зуммер - требует внешнего генератора сигнала (функция tone())

Активный зуммер - имеет встроенный генератор, звучит при подаче напряжения

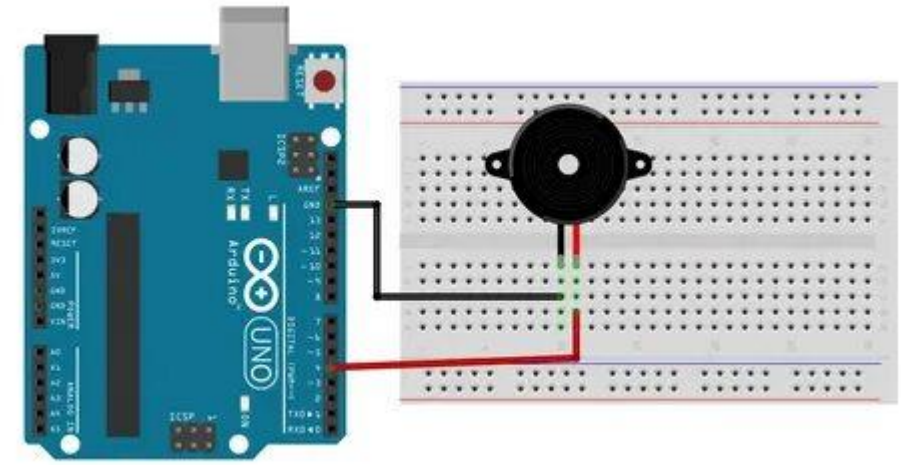

Рисунок 5 – Зуммер

Подключение зуммера

Зуммер (+ → Пин Arduino)

Зуммер (- → GND)

Рекомендуется: Добавить токоограничивающий резистор 100-220 Ом последовательно с зуммером.

Генерация звука с помощью tone()

tone(pin, frequency); // Включить звук заданной частоты

tone(pin, frequency, duration); // Включить звук на определенное время

noTone(pin); // Выключить звук

Диапазоны частот:

Человеческое ухо: 20 Гц - 20 кГц

Оптимальный диапазон для зуммера: 500 Гц - 5 кГц

Ноты музыкальной октавы: 262 Гц (C4) - 523 Гц (C5)

Пример генерации мелодии

```
// Частоты нот для октавы
#define NOTE_C4  262
#define NOTE_D4  294
#define NOTE_E4  330
```

```
#define NOTE_F4  349
#define NOTE_G4  392
#define NOTE_A4  440
#define NOTE_B4  494
#define NOTE_C5  523

void playNote(int pin, int note, int duration) {
  tone(pin, note, duration);
  delay(duration);        // Ждем пока нота звучит
  delay(50);          // Короткая пауза между нотами
  noTone(pin);
}
```

Обработка множества кнопок

```
// Опрос нескольких кнопок
for (int i = 0; i < numButtons; i++) {
  if (digitalRead(buttonPins[i]) == LOW) {
    // Обработка нажатия кнопки i
    delay(50); // Антидребезг
  }
}
```

Определение длительности нажатия

```
unsigned long pressTime = 0;

void checkLongPress(int pin) {
  if (digitalRead(pin) == LOW) {
    if (pressTime == 0) {
      pressTime  =  millis();  //  Запоминаем  время
начала нажатия
    }
    // Проверяем длительность нажатия
    if (millis() - pressTime > 3000) {
      Serial.println("Долгое нажатие!");
      pressTime = 0; // Сбрасываем таймер
    }
  } else {
    pressTime = 0; // Кнопка отпущена
  }
}
```

Управление звуком без блокирующих задержек

```
unsigned long previousNoteTime = 0;
int currentNote = 0;

void playNonBlocking(int melody[], int durations[],
int length) {
  unsigned long currentTime = millis();

  if    (currentTime    -    previousNoteTime    >=
durations[currentNote]) {
    // Время играть следующую ноту
    tone(buzzerPin, melody[currentNote]);
    currentNote = (currentNote + 1) % length;
    previousNoteTime = currentTime;
  }
}
```

ТИПИЧНЫЕ ПРОБЛЕМЫ И РЕШЕНИЯ

Проблема 1: Фантомные нажатия
Решение: использовать подтягивающие резисторы (INPUT_PULLUP)

Проблема 2: Дребезг при нажатии
Решение: добавить программную задержку или аппаратный фильтр

Проблема 3: Слабый звук зуммера
Решение:
- убедиться, что используется пассивный зуммер
- проверить подключение (+ к пину, - к GND)
- увеличить частоту (в разумных пределах)

Проблема 4: Кнопка не реагирует
Решение:
- проверить подключение (пин → кнопка → GND)
- убедиться, что используется INPUT_PULLUP
- проверить логику (LOW при нажатии)

НЕОБХОДИМО ВОСПРОИЗВЕСТИ МОНТАЖНУЮ СХЕМУ И СДЕЛАТЬ ПРИНЦИПИАЛЬНУЮ СХЕМУ В ПРОГРАММЕ FRITZING

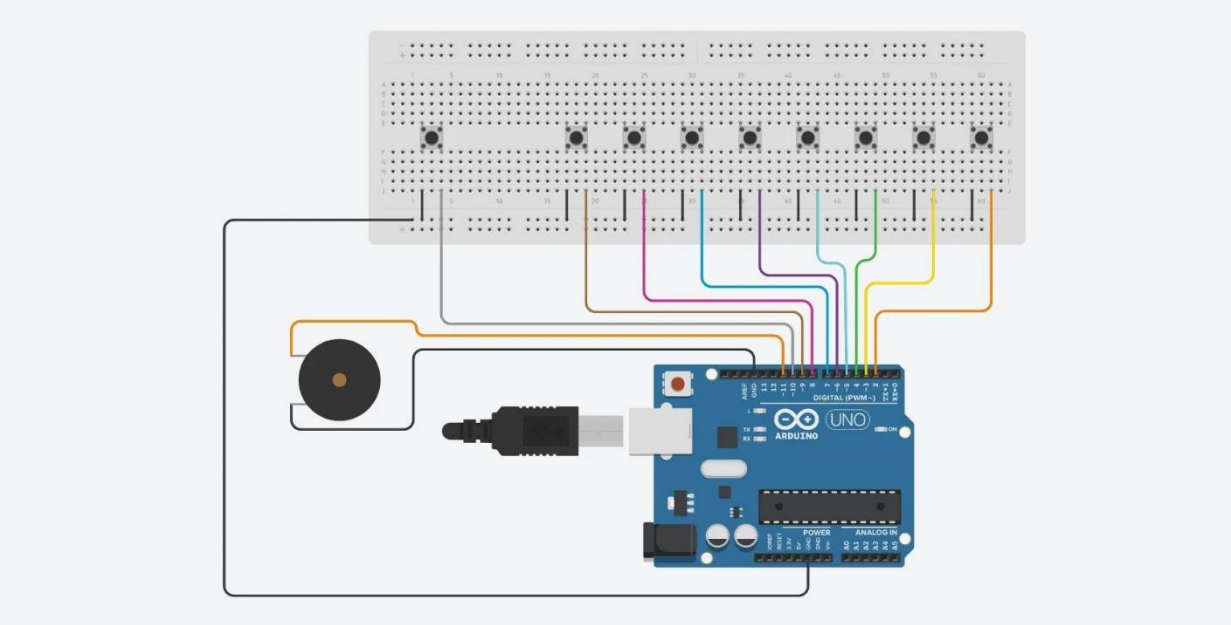

Рисунок 5 – Схема подключения

Подключение компонентов

Каждый светодиод подключен анодом (+) к пину 2-13 через резистор 220 Ом, катодом (–) к GND. Резистор необходим в схеме для предотвращения перегорания светодиода.

ЗАДАНИЯ

Программа 1: Пианино на 8 нот

Кнопки на пинах 2-9 (INPUT_PULLUP) соответствуют нотам от C4 до C5. При нажатии кнопки звук воспроизводится через зуммер на пине 11.

```
// Частоты нот для октавы C4-C5 (в Гц)
int notes[] = {262, 294, 330, 349, 392, 440, 494, 523};

void setup() {
  Serial.begin(9600);
  // Настраиваем пины 2-9 на вход с подтяжкой к VCC
  for (int i = 2; i <= 9; i++) {
    pinMode(i, INPUT_PULLUP);
  }
  pinMode(11, OUTPUT); // Зуммер
  Serial.println("Пианино готово! Нажмите кнопки 2-9.");
}

void loop() {

  // Проверяем каждую кнопку
  for (int i = 0; i < 8; i++) {
    int buttonPin = i + 2; // Кнопки на пинах 2-9
    if (digitalRead(buttonPin) == LOW) {
// Если кнопка нажата
      tone(11, notes[i]); // Воспроизводим ноту
      Serial.print("Играет нота: ");
      Serial.println(notes[i]);
      delay(200); // Задержка для антидребезга
      while (digitalRead(buttonPin) == LOW); // Ждем
отпускания кнопки
      noTone(11); // Выключаем звук
    }
  }
}
```

Программа 2: Запись и воспроизведение

Кнопка 10:
- Короткое нажатие: старт/стоп записи.
- Удержание 3 сек: воспроизведение записи.
- Во время воспроизведения: любое нажатие — пауза, удержание 3 сек — сброс.

```
// Частоты нот для октавы C4-C5 (в Гц)
int notes[] = {262, 294, 330, 349, 392, 440, 494, 523};

#define RECORD_PIN 10
#define BUZZER_PIN 11
#define NUM_NOTES 8

// Массивы для хранения записи (ноты и временные интервалы)
int recordedNotes[100];
unsigned long recordedTimes[100];
int noteCount = 0;
bool isRecording = false;
bool isPlaying = false;
bool isPaused = false;

unsigned long pressStartTime = 0;
bool wasPressed = false;

void setup() {
  Serial.begin(9600);
  for (int i = 2; i <= 9; i++) pinMode(i, INPUT_PULLUP);
  pinMode(RECORD_PIN, INPUT_PULLUP);
  pinMode(BUZZER_PIN, OUTPUT);
  Serial.println("Режим записи. Нажмите 10 для старта, еще
раз для остановки.");
}

void loop() {
  handleRecordButton(); // Обработка кнопки
записи/воспроизведения
  if (isRecording) {
    recordNotes(); // Запись нот
  } else if (isPlaying && !isPaused) {
    playRecordedNotes(); // Воспроизведение
  }
}
```

```
void handleRecordButton() {
  int buttonState = digitalRead(RECORD_PIN);
  if (buttonState == LOW && !wasPressed) {
    pressStartTime = millis(); // Засекаем время нажатия
    wasPressed = true;
  }

  if (buttonState == HIGH && wasPressed) {
    unsigned long pressDuration = millis() - pressStartTime;
    if (pressDuration < 3000) {
      // Короткое нажатие
      if (!isPlaying) {
        toggleRecording(); // Старт/стоп записи
      } else {
        togglePause(); // Пауза при воспроизведении
      }
    }
    wasPressed = false;
  }

  // Проверка удержания 3 секунды
  if (wasPressed && (millis() - pressStartTime >= 3000)) {
    if (!isRecording && !isPlaying) {
      startPlayback(); // Начало воспроизведения
    } else if (isPlaying) {
      resetPlayback(); // Сброс воспроизведения
    }
    wasPressed = false;
  }
}

void toggleRecording() {
  isRecording = !isRecording;
  if (isRecording) {
    noteCount = 0; // Сброс предыдущей записи
    Serial.println("Запись начата...");
  } else {
    Serial.println("Запись остановлена.");
  }
}

void recordNotes() {
  for (int i = 0; i < NUM_NOTES; i++) {
    if (digitalRead(i + 2) == LOW) {
```

```
      tone(BUZZER_PIN, notes[i]);
      recordedNotes[noteCount] = notes[i];
      recordedTimes[noteCount] = millis();
      noteCount++;
      Serial.print("Записано: ");
      Serial.println(notes[i]);
      delay(200);
      while (digitalRead(i + 2) == LOW); // Ждем отпускания
      noTone(BUZZER_PIN);
      break;
    }
  }
}

void startPlayback() {
  if (noteCount > 0) {
    isPlaying = true;
    isPaused = false;
    Serial.println("Воспроизведение...");
  }
}

void togglePause() {
  isPaused = !isPaused;
  Serial.println(isPaused ? "Пауза" : "Продолжено");
}

void resetPlayback() {
  isPlaying = false;
  isPaused = false;
  Serial.println("Сброс воспроизведения.");
}

void playRecordedNotes() {
  for (int i = 0; i < noteCount; i++) {
    if (!isPlaying || isPaused) break;
    tone(BUZZER_PIN, recordedNotes[i]);
    Serial.print("Воспроизведение: ");
    Serial.println(recordedNotes[i]);
    // Вычисляем длительность ноты из разницы времен
    unsigned long noteDuration = (i < noteCount - 1) ?
recordedTimes[i + 1] - recordedTimes[i] : 500;
    delay(noteDuration);
    noTone(BUZZER_PIN);
```

```
  }
  isPlaying = false;
  Serial.println("Воспроизведение завершено.");
}
```

Программа 3: Воспроизведение трех мелодий

Плеер с тремя мелодиями. Кнопка на пине 10 переключает мелодии: Jingle Bells, Happy Birthday, We Are.

```
// Определяем частоты нот (в Герцах)
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
```

```
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
```

```
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

// Пауза
#define NOTE_PAUSE 0

// Длительности нот (в миллисекундах)
#define WHOLE_NOTE 1600
#define HALF_NOTE 800
#define QUARTER_NOTE 400
#define EIGHTH_NOTE 200
#define SIXTEENTH_NOTE 100

// МЕЛОДИЯ 1: JINGLE BELLS
int jingleBellsNotes[] = {
  NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4,
  NOTE_E4, NOTE_G4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_PAUSE,
  NOTE_F4, NOTE_F4, NOTE_F4, NOTE_F4, NOTE_F4, NOTE_E4,
  NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_D4, NOTE_D4,
  NOTE_E4, NOTE_D4, NOTE_G4, NOTE_PAUSE,
  NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4, NOTE_E4,
  NOTE_E4, NOTE_G4, NOTE_C4, NOTE_D4, NOTE_E4, NOTE_PAUSE,
  NOTE_F4, NOTE_F4, NOTE_F4, NOTE_F4, NOTE_F4, NOTE_E4,
  NOTE_E4, NOTE_E4, NOTE_G4, NOTE_G4, NOTE_F4, NOTE_D4, NOTE
_C4
};

int jingleBellsDurations[] = {
  QUARTER_NOTE, QUARTER_NOTE, HALF_NOTE, QUARTER_NOTE, QUART
ER_NOTE, HALF_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, WH
OLE_NOTE, QUARTER_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QU
ARTER_NOTE, QUARTER_NOTE,
```

```
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QU
ARTER_NOTE, QUARTER_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, WHOLE_NOTE, QUARTER_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, HALF_NOTE, QUARTER_NOTE, QUART
ER_NOTE, HALF_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, WH
OLE_NOTE, QUARTER_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QU
ARTER_NOTE, QUARTER_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, HA
LF_NOTE, QUARTER_NOTE, WHOLE_NOTE
};

int jingleBellsLength = 53;

// МЕЛОДИЯ 2: HAPPY BIRTHDAY
int happyBirthdayNotes[] = {
  NOTE_C4, NOTE_C4, NOTE_D4, NOTE_C4, NOTE_F4, NOTE_E4, NOTE
_PAUSE,
  NOTE_C4, NOTE_C4, NOTE_D4, NOTE_C4, NOTE_G4, NOTE_F4, NOTE
_PAUSE,
  NOTE_C4, NOTE_C4, NOTE_C5, NOTE_A4, NOTE_F4, NOTE_E4, NOTE
_D4, NOTE_PAUSE,
  NOTE_AS4, NOTE_AS4, NOTE_A4, NOTE_F4, NOTE_G4, NOTE_F4
};

int happyBirthdayDurations[] = {
  EIGHTH_NOTE, EIGHTH_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUAR
TER_NOTE, HALF_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUAR
TER_NOTE, HALF_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUAR
TER_NOTE, QUARTER_NOTE, HALF_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUAR
TER_NOTE, WHOLE_NOTE
};

int happyBirthdayLength = 28;

// МЕЛОДИЯ 3: WE ARE
int weAreNotes[] = {
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
```

```
  NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,

  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,

  NOTE_E4, NOTE_G4, NOTE_C5, NOTE_E5,
  NOTE_D5, NOTE_C5, NOTE_G4,
  NOTE_E4, NOTE_G4, NOTE_C5, NOTE_E5,
  NOTE_D5, NOTE_C5, NOTE_G4,

  NOTE_E4, NOTE_G4, NOTE_C5, NOTE_E5,
  NOTE_D5, NOTE_C5, NOTE_A4,
  NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4,
  NOTE_C4, NOTE_PAUSE,

  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_C5, NOTE_G4, NOTE_E4,
  NOTE_G4, NOTE_F4, NOTE_E4, NOTE_D4
};

int weAreDurations[] = {
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,

  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,

  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE,
  HALF_NOTE, HALF_NOTE, HALF_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE,
  HALF_NOTE, HALF_NOTE, HALF_NOTE,

  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE,
  HALF_NOTE, HALF_NOTE, HALF_NOTE,
  QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE, QUARTER_NOTE,
  WHOLE_NOTE, QUARTER_NOTE,
```

```
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE,
  EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE, EIGHTH_NOTE
};

int weAreLength = 68;

// Массивы для доступа к мелодиям
int* melodies[] = {jingleBellsNotes, happyBirthdayNotes, weA
reNotes};
int* durations[] = {jingleBellsDurations, happyBirthdayDurat
ions, weAreDurations};
int melodyLengths[] = {jingleBellsLength, happyBirthdayLengt
h, weAreLength};

// Переменные для управления воспроизведением
int currentSong = 0;
bool isPlaying = false;
int currentNote = 0;
unsigned long previousNoteTime = 0;
unsigned long currentNoteDuration = 0;

// Пины
const int BUTTON_PIN = 10;
const int BUZZER_PIN = 11;

void setup() {
  Serial.begin(9600);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(BUZZER_PIN, OUTPUT);

  Serial.println("=== ПЛЕЕР АРДУИНО ===");
  Serial.println("Нажмите кнопку на пине 10 для начала воспр
оизведения");
  Serial.println("Доступные мелодии:");
  Serial.println("1. Jingle Bells");
  Serial.println("2. Happy Birthday");
  Serial.println("3. We Are ");
}

void loop() {
  // Обработка кнопки переключения мелодий
  handleButton();
```

```
  // Воспроизведение текущей мелодии
  if (isPlaying) {
    playMusic();
  }
}

void handleButton() {
  static unsigned long lastButtonPress = 0;
  static bool buttonPressed = false;

  if (digitalRead(BUTTON_PIN) == LOW) {
    if (!buttonPressed && (millis() - lastButtonPress > 300)
) {
      // Обработка нажатия кнопки
      buttonPressed = true;
      lastButtonPress = millis();

      if (!isPlaying) {
        // Начало воспроизведения
        isPlaying = true;
        currentNote = 0;
        Serial.println("\n=== НАЧАЛО ВОСПРОИЗВЕДЕНИЯ ===");
        printCurrentSongInfo();
      } else {
        // Переключение на следующую песню
        currentSong = (currentSong + 1) % 3;
        currentNote = 0;
        noTone(BUZZER_PIN);
        Serial.println("\n=== ПЕРЕКЛЮЧЕНИЕ ПЕСНИ ===");
        printCurrentSongInfo();
      }
    }
  } else {
    buttonPressed = false;
  }
}

void playMusic() {
  unsigned long currentTime = millis();

  // Если время текущей ноты истекло
  if (currentTime - previousNoteTime >= currentNoteDuration)
{
```

```
    // Останавливаем предыдущую ноту
    noTone(BUZZER_PIN);

    // Проверяем, не закончилась ли мелодия
    if (currentNote >= melodyLengths[currentSong]) {
      // Мелодия закончилась
      isPlaying = false;
      Serial.println("=== ВОСПРОИЗВЕДЕНИЕ ЗАВЕРШЕНО ===");
      return;
    }

    // Получаем следующую ноту и ее длительность
    int note = melodies[currentSong][currentNote];
    currentNoteDuration = durations[currentSong][currentNote
];

    // Воспроизводим ноту (если это не пауза)
    if (note != NOTE_PAUSE) {
      tone(BUZZER_PIN, note);
      Serial.print("Нота: ");
      Serial.print(note);
      Serial.print(" Гц, длительность: ");
      Serial.print(currentNoteDuration);
      Serial.println(" мс");
    } else {
      Serial.println("Пауза");
    }

    // Переходим к следующей ноте
    currentNote++;
    previousNoteTime = currentTime;
  }
}

void printCurrentSongInfo() {
  Serial.print("Сейчас играет: ");
  switch (currentSong) {
    case 0:
      Serial.println("Jingle Bells");
      Serial.println("(Полная версия)");
      break;
    case 1:
      Serial.println("Happy Birthday");
      Serial.println("(Полная версия)");
```

```
      break;
    case 2:
      Serial.println("We Are");
      Serial.println("(Полная версия)");
      break;
  }
  Serial.print("Количество нот: ");
  Serial.println(melodyLengths[currentSong]);
  Serial.println("----------------------------");
}
```