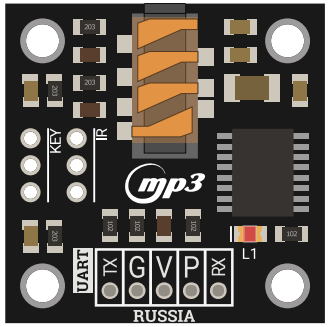
MP3-плеер (Trema-модуль v2.0)

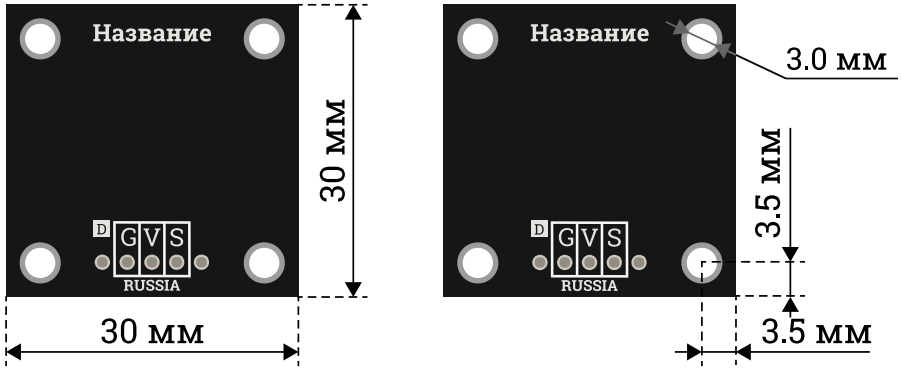


Общие сведения:

Trema-модуль MP3-плеер — MP3-проигрыватель, управляемый кнопками или при помощи Arduino, который вы можете использовать в широком диапазоне задач.

Спецификация:

- Поддержка форматов: MP3,WMV;
- 24-разрядный ЦАП
- Поддерживаемые карты памяти: SD-карты (FAT16/FAT32);
- Объём карты памяти: SD до 2Gb / SDHC до 32Gb;
- Питание: 3.3 - 5В;
- Наличие режима "Рекламная пауза", "Непрерывное воспроизведение", "Случайный порядок";
- Глубина архива: до 100 папок, каждая папка может содержать до 255 песен;
- 30 уровней громкости;
- 6 уровней EQ (Эквалайзера);
- Все модули линейки "Trema" выполнены в одном формате



Описание выводов:

RX (Принял)	Вывод RX универсального асинхронного приёмопередатчика (Подключается к выводу TX Arduino)
TX (Передал)	Вывод TX универсального асинхронного приёмопередатчика (Подключается к выводу RX Arduino)
P	Вывод статуса проигрывания (LOW - идёт воспроизведение, HIGH - воспроизведение остановлено)
V	Напряжение питания +3,3 ... +5 вольт постоянного тока
G	Напряжение питания 0 В (земля, общий, GND...)

Подключение:

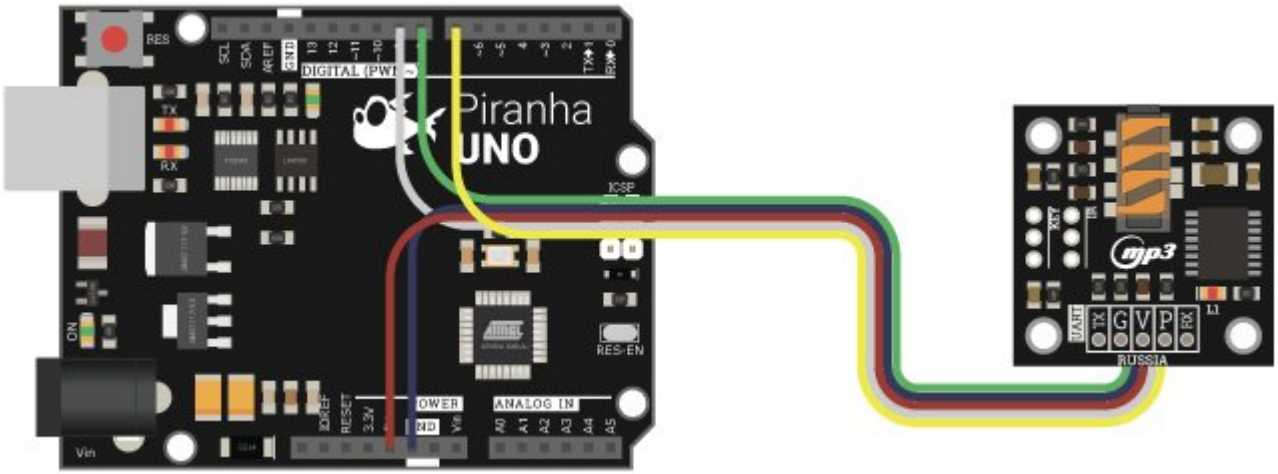
Для удобства подключения к Arduino воспользуйтесь Trema Shield, Trema Power Shield, Motor Shield или Trema Set Shield.

Выводы MP3-плеера	**Выводы Arduino**
RX	TX (9 в примерах)
TX	RX (8 в примерах)
P	Любой вывод
G	GND
V	5V

Модуль удобно подключать 3 способами, в зависимости от ситуации:

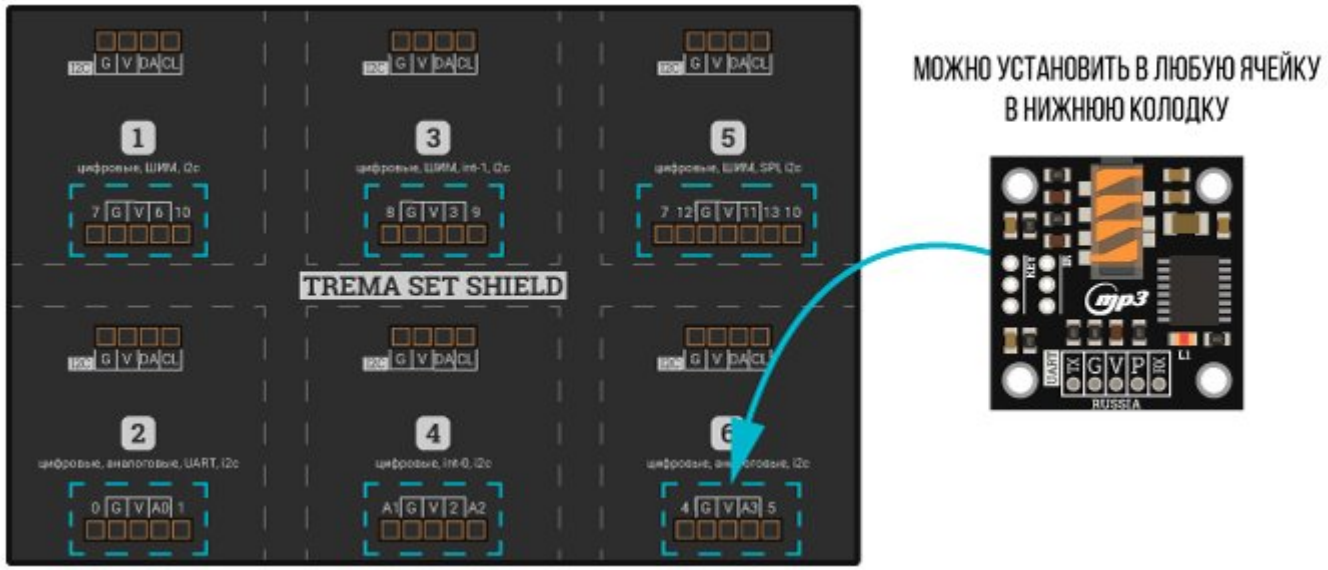
Способ - 1 : Используя проводной шлейф и Piranha UNO

Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO



Способ - 2 : Используя Trema Set Shield

Модуль можно подключить к любому из входов Trema Set Shield (при использовании программного последовательного порта, указав соответствующие выводы для RX и TX)

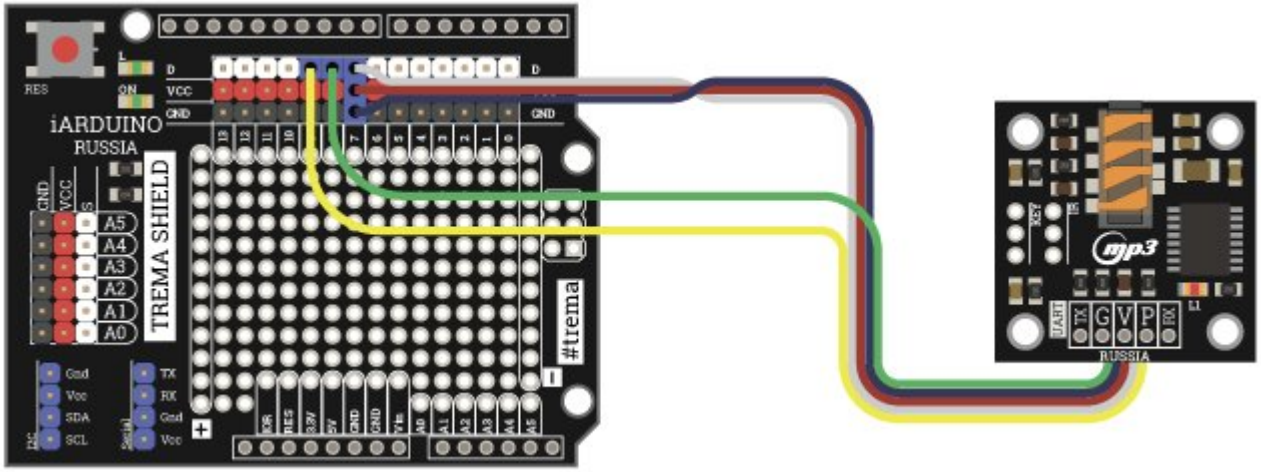


Для аппаратной шины UART (на выводах 0, 1), модуль устанавливается во 2 ячейку.

Для аппаратной шины UART (на выводах 8, 9), модуль устанавливается в 3 ячейку.

Способ - 3 : Используя проводной шлейф и Shield

Используя 5-и проводной шлейф, к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.



Питание:

Данный модуль питается от источника постоянного напряжения 3.3В или 5В.

Подробнее о модуле:

Модуль имеет удобный 3,5мм аудио-выход, разъём для подключения карты памяти и клавиатуры, удобную колодку для подключения к Arduino, а так же светодиод, который информирует о состоянии плеера в текущий момент - находится он в режиме ожидания или воспроизведения. При проигрывании файлов светодиод горит.

Модуль прекрасно воспроизводит самые популярные форматы (MP3, WAV) и имеет целый ряд программных возможностей: 6 предустановленных настроек эквалайзера, режим зацикливания и случайного воспроизведения, регулировку громкости, номера и папки с воспроизводимыми треками.

На модуле имеется вывод "**P**", который соединён параллельно светодиоду состояния, благодаря чему можно считывать с этого вывода логический уровень сигнала и использовать его в дальнейших сценариях (LOW - идёт воспроизведение, HIGH - воспроизведение остановлено).

Для работы с модулем мы рекомендуем использовать библиотеку **DFPlayerMini**, которую вы можете скачать с нашего сайта.

Структура папок.

Данный модуль имеет ряд особенностей, связанных с построением каталога воспроизводимых файлов и папок. Об этом мы сейчас и поговорим.

Плеер поддерживает 3 типа внутренней структуры папок:

- **Папки, пронумерованные от 01 до 99 - стандартное воспроизведение треков:**
  - Папки, пронумерованные от 01 до 99 могут включать в себя не более 255 файлов каждая;
  - Папки, пронумерованные от 01 до 10 могут включать в себя не более 1000 файлов каждая;
- **Папка MP3 - стандартное воспроизведение треков:**
  - Может включать в себя 65535 файлов, пронумерованных от 001 до 65535;
  - Папка должна находится в корневом каталоге
- **Папка ADVERT - рекламное воспроизведение треков:**
  - Может включать в себя 65535 файлов, пронумерованных от 001 до 65535;
  - Папка должна находится в корневом каталоге

Работа с файлами.

Работа с файлами предполагает, что все они были переименованы согласно тому диапазону, который был задан типом структуры папок. Плеер будет проигрывать файлы и с буквами, но при этом ухудшается удобство навигации по файлам. Самый простой способ для удобства навигации - добавить цифры в начале файла, например: 001 - Cannibal Corpse - I Will Kill You.mp3, 002 - Nile - Call to Destruction.mp3 и т. д. Ниже приведём простой пример Bash-сценария и Cmd-сценария для быстрого переименовывания файлов.

Bash-сценарий для быстрого переименовывания файлов (GNU/Linux, macOS):

- Создайте файл rename.sh

- Скопируйте или введите с клавиатуры нижеприведённый код в этот файл
- В эмуляторе теримнала сделайте файл запускаемым: **chmod +x rename.sh**
- Запускайте в директории с файлами, которые необходимо переименовать

```
#!/bin/bash
j=1
for i in *.mp3
    do
        if [[ $j -lt 10 ]]
            then x="00$j"
        elif [[ $j -lt 100 ]]
            then x="0$j"
        else
            x="$j"
        fi
        newname="$x - $i"
        echo $i" -> "$newname
        mv $i "${newname}"
        let "j++"
    done
```

Cmd-сценарий для быстрого переименовывания файлов (Windows):

- Создайте файл rename.bat
- Скопируйте или введите с клавиатуры нижеприведённый код в этот файл
- Запускайте в папке с файлами, которые необходимо переименовать

```
@echo off
setlocal enableDelayedExpansion
set /a n = 1
for /f %%f in ('dir /B/D .\*.mp3') do (
        if !n! LSS 10 (
```

```
                set x=00!n!
        ) else (
                if !n! LSS 100 (
                        set x=0!n!
                ) else (
                        set x=!n!
                        )
        )
        echo %%f -^> !x! - %%f
        mv "%%f" "!x! - %%f"
        set /a n+=1
)
```

Воспроизведение.

Данный плеер при стандартных настройках устанавливает однократное воспроизведение файлов. В этом случае вы можете определить окончание трека либо с помощью встроенной функции `readState()`, либо по уровню сигнала на **выводе P**, который подключен параллельно светодиодной индикации плеера и имеет на выходе обратный логический уровень светодиода:

- **горит** - плеер находится в режиме воспроизведения (на выводе логический ноль, LOW)
- **не горит** - плеер находится в режиме паузы или прекращения проигрывания (на выводе логическая единица, HIGH)

Данным выводом удобно пользоваться, если вы хотите отслеживать состояние переключения трека, которое происходит слишком быстро, чтобы всегда успеть определить это в коде программы. Используя прерывания и логический уровень вывода P вы сможете избежать этой проблемы.

Примеры:

Поочерёдное использование основных функций библиотеки.

```
#include <SoftwareSerial.h>                                   // Подключаем библиотеку для работы
#include <DFRobotDFPlayerMini.h>                              // Подключаем библиотеку для работы
```

```
#define RX 8                                                    // Определяем вывод RX (TX на плеере
#define TX 9                                                    // Определяем вывод TX (RX на плеере
SoftwareSerial mySoftwareSerial(RX, TX);                        // создаём объект mySoftwareSerial и
DFRobotDFPlayerMini myDFPlayer;                                 // создаём объект myDFPlayer для раб

void printDetail(uint8_t type, int value);                      // создаём функцию для опроса плеера

uint32_t timer = millis();                                      // создаём счётчик для выполнения фу

void setup() {
  mySoftwareSerial.begin(9600);                                 // инициализируем работу с программн
  Serial.begin(115200);                                         // инициализируем работу с аппаратны
  Serial.println();                                             // Выводим в монитор порта пустую ст
  Serial.println(F("DFRobot DFPlayer Mini Demo"));              // Выводим текст в монитор последова
  Serial.println(F("Initializing DFPlayer..."));                // Выводим текст в монитор последова
  if (!myDFPlayer.begin(mySoftwareSerial)) {                    // Проверяем, есть ли связь с плеерс
    Serial.println(F("Unable to begin:"));                      // Выводим текст в монитор последова
    Serial.println(F("1.Please recheck the connection!"));      // Выводим текст в монитор последова
    Serial.println(F("2.Please insert the SD card!"));          // Выводим текст в монитор последова
    while (true);                                               // Скетч не выполняется дальше, если
  }
  Serial.println(F("DFPlayer Mini online."));                   // Выводим текст в монитор последова
  myDFPlayer.setTimeOut(500);                                   // Указываем время отклика плеера на

  // НАСТРОЙКИ ЗВУКА
  myDFPlayer.volume(10);                                        // Устанавливаем громкость равной 10
  myDFPlayer.volumeUp();                                        // Увеличить громкость на 1
  myDFPlayer.volumeDown();                                      // Уменьшить громкость на 1

  // НАСТРОЙКИ ЭКВАЛАЙЗЕРА
  myDFPlayer.EQ(DFPLAYER_EQ_NORMAL);                            // Устанавливаем эквалайзер в положе
  //  myDFPlayer.EQ(DFPLAYER_EQ_POP);                           // Устанавливаем эквалайзер в положе
  //  myDFPlayer.EQ(DFPLAYER_EQ_ROCK);                          // Устанавливаем эквалайзер в положе
```

```
  //  myDFPlayer.EQ(DFPLAYER_EQ_JAZZ);                    // Устанавливаем эквалайзер в положе
  //  myDFPlayer.EQ(DFPLAYER_EQ_CLASSIC);                 // Устанавливаем эквалайзер в положе
  //  myDFPlayer.EQ(DFPLAYER_EQ_BASS);                    // Устанавливаем эквалайзер в положе

  // НАСТРОЙКИ ИСТОЧНИКА ФАЙЛОВ
  //  myDFPlayer.outputDevice(DFPLAYER_DEVICE_U_DISK);    // Устанавливаем источником USB-ДИСК
  myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);           // Устанавливаем источником SD-карту
  //  myDFPlayer.outputDevice(DFPLAYER_DEVICE_AUX);       // Устанавливаем источником AUX
  //  myDFPlayer.outputDevice(DFPLAYER_DEVICE_SLEEP);     // Устанавливаем режим сна
  //  myDFPlayer.outputDevice(DFPLAYER_DEVICE_FLASH);     // Устанавливаем источником FLASH-ка

  // НАСТРОЙКИ РАБОТЫ ПЛЕЕРА
  //  myDFPlayer.sleep();                                 // Установить режим СНА
  //  myDFPlayer.reset();                                 // Сбросить все настройки
  //  myDFPlayer.enableDAC();                             // Включить ЦАП на чипе
  //  myDFPlayer.disableDAC();                            // Выключить ЦАП на чипе
  //  myDFPlayer.outputSetting(true, 15);                 // Настройки усилителя - включить и

  // НАСТРОЙКИ ВОСПРОИЗВЕДЕНИЯ
  myDFPlayer.next();                                     // Включить воспроизведение следующе
  delay(3000);                                           // Задержка в 3 секунды
  myDFPlayer.previous();                                 // Включить воспроизведение предыду
  delay(3000);                                           // Задержка в 3 секунды
  myDFPlayer.play(1);                                    // Включить воспроизведение указанно
  delay(3000);                                           // Задержка в 3 секунды
  myDFPlayer.loop(1);                                    // Включить неперывное воспроизведен
  delay(3000);                                           // Задержка в 3 секунды
  myDFPlayer.pause();                                    // Поставить воспроизведение на пауз
  delay(3000);                                           // Задержка в 3 секунды
  myDFPlayer.start();                                    // Продолжить воспроизведение трека
  delay(3000);                                           // Задержка в 3 секунды
  myDFPlayer.playFolder(15, 4);                          // Включить воспроизведение указанно
```

```
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.enableLoopAll();                                     // Включить непрерывное воспроизведен
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.disableLoopAll();                                    // Выключить непрерывное воспроизведе
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.playMp3Folder(4);                                    // Включить воспроизведение указанно
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.advertise(3);                                        // Включить воспроизведение указанно
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.stopAdvertise();                                     // Выключить воспроизведение трека в
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.playLargeFolder(2, 999);                             // Включить воспроизведение указанно
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.loopFolder(5);                                       // Включить непрерывное воспроизведен
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.randomAll();                                         // Включить случайное воспроизведени
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.enableLoop();                                        // Включить непрерывное воспроизведен
  delay(3000);                                                    // Задержка в 3 секунды
  myDFPlayer.disableLoop();                                       // Выключить непрерывное воспроизведе
  delay(3000);                                                    // Задержка в 3 секунды

  Serial.println(myDFPlayer.readState());                         // Выводим текст: "Состояние проигры
  Serial.println(myDFPlayer.readVolume());                        // Выводим текст: "Значение установл
  Serial.println(myDFPlayer.readEQ());                            // Выводим текст: "Тип установленног
  Serial.println(myDFPlayer.readFileCounts());                    // Выводим текст: "Количество файлов
  Serial.println(myDFPlayer.readCurrentFileNumber());             // Выводим текст: "Номер проигрываем
  Serial.println(myDFPlayer.readFileCountsInFolder(3));           // Выводим текст: "Количество файлов
}

void loop() {
  if (millis() - timer > 3000) {                                  // Раз в 3 секнуды выполняем:
    timer = millis();                                             // Обнуление счётчика
```

```
    myDFPlayer.next();                                                    // Переключение трека на следующий
  }
  if (myDFPlayer.available()) {                                           // Если плеер на связи, то
    printDetail(myDFPlayer.readType(), myDFPlayer.read());                // Выводим подробный текст с описани
  }
}

void printDetail(uint8_t type, int value) {                               // функция опроса плеера на предмет
  switch (type) {                                                         // в зависимости от полученного знач
    case TimeOut:               Serial.println(F("Time Out!"));               break;    // Выводим текст: "Время ожидания ис
    case WrongStack:            Serial.println(F("Stack Wrong!"));            break;    // Выводим текст: "Неверный стек"
    case DFPlayerCardInserted: Serial.println(F("Card Inserted!"));           break;    // Выводим текст: "Карта не читается
    case DFPlayerCardRemoved:  Serial.println(F("Card Removed!"));            break;    // Выводим текст: "Карта не установл
    case DFPlayerCardOnline:   Serial.println(F("Card Online!"));             break;    // Выводим текст: "Карта установлена
    case DFPlayerUSBInserted:  Serial.println("USB Inserted!");               break;    // Выводим текст: "USB установлено"
    case DFPlayerUSBRemoved:   Serial.println("USB Removed!");               break;    // Выводим текст: "USB отключено"
    case DFPlayerPlayFinished: Serial.print(F("Number:"));                                // Выводим текст: "Номер трека:"
                               Serial.print(value);                                      // Выводим значение value - номер тр
                               Serial.println(F(" Play Finished!"));          break;    // Выводим текст: "закончился"
    case DFPlayerError:        Serial.print(F("DFPlayerError:"));                        // Выводим текст: "Список ошибок"
      switch (value) {                                                                   //
        case Busy:             Serial.println(F("Card not found"));          break;     // Выводим текст: "Карта не найдена"
        case Sleeping:         Serial.println(F("Sleeping"));                break;     // Выводим текст: "Режим сна"
        case SerialWrongStack: Serial.println(F("Get Wrong Stack"));         break;     // Выводим текст: "Получен неверный
        case CheckSumNotMatch: Serial.println(F("Check Sum Not Match"));     break;     // Выводим текст: "Неправильная конт
        case FileIndexOut:     Serial.println(F("File Index Out of Bound")); break;     // Выводим текст: "Номер трека превы
        case FileMismatch:     Serial.println(F("Cannot Find File"));        break;     // Выводим текст: "Не найден файл"
        case Advertise:        Serial.println(F("In Advertise"));            break;     // Выводим текст: "В режиме рекламно
        default:                                                             break;     //
      }                                                                        break;    //
    default:                                                                   break;    //
  }
}
```

Описание функций библиотеки:

Подключение библиотеки:

```
#include <SoftwareSerial.h>              // Подключаем библиотеку для работы с последовательным интерфейсом
#include <DFRobotDFPlayerMini.h>         // Подключаем библиотеку для работы с плеером
#define RX 8                             // Определяем вывод RX (TX на плеере) программного последовательного порта
#define TX 9                             // Определяем вывод TX (RX на плеере) программного последовательного порта
SoftwareSerial mySoftwareSerial(RX, TX); // создаём объект mySoftwareSerial и указываем выводы, к которым подключен плеер (RX,
DFRobotDFPlayerMini myDFPlayer;          // создаём объект myDFPlayer для работы с плеером
```

Функция begin();

- Назначение: инициирование работы плеера;
- Синтаксис: **begin();**
- Параметры: Нет;
- Возвращаемые значения: Нет;
- Примечание:
  - Функцию необходимо вызвать до обращения к любым другим функциям библиотеки;
  - Функцию достаточно вызвать один раз в коде `setup` ;
  - Функцию можно использовать для определения наличия плеера;
- Пример:

```
myDFPlayer.begin(mySoftwareSerial); // инициализируем работу плеера
```

Функция available();

- Назначение: проверка наличия связи с плеером;

- Синтаксис: **available()**;
- Параметры: нет
- Возвращаемые значения:
- **true** - если есть связь / **false** - если связи нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.available(); // Запрос на наличие связи с плеером
```

Функция setTimeOut();

- Назначение: установка времени отклика плеера;
- Синтаксис: **setTimeOut(ПАРАМЕТР)**;
- Параметры:
  - **ПАРАМЕТР** - время отклика в мс;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.setTimeOut(500); // Устанавливаем время отклика плеера равным 500мс
```

Функция volume();

- Назначение: установка уровня громкости;
- Синтаксис: **volume(ПАРАМЕТР)**;
- Параметры:
  - **ПАРАМЕТР** - значение уровня громкости (от 0 до 30);
- Возвращаемые значения: нет;
- Примечание:
- Пример:

```
myDFPlayer.volume(10); // Устанавливаем громкость равной 10 (от 0 до 30)
```

Функция volumeUp;

- Назначение: увеличение уровня громкости на 1 значение;
- Синтаксис: **volumeUp()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.volumeUp(); // Увеличить громкость на 1
```

Функция volumeDown();

- Назначение: уменьшение уровня громкости на 1 значение;
- Синтаксис: **volumeDown()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.volumeDown(); // Уменьшить громкость на 1
```

Функция EQ();

- Назначение: выбор одного из 6 предустановленных эквалайзеров;
- Синтаксис: **EQ(ПАРАМЕТР)**;
- Параметры:
  - **ПАРАМЕТР** - один из 6 типов предустановленных настроек эквалайзера:

        - **DFPLAYER_EQ_NORMAL;**
        - **DFPLAYER_EQ_POP;**
        - **DFPLAYER_EQ_ROCK;**
        - **DFPLAYER_EQ_JAZZ;**
        - **DFPLAYER_EQ_CLASSIC;**
        - **DFPLAYER_EQ_BASS;**
- Возвращаемые значения: нет;
- Примечание:
- Пример:

```
myDFPlayer.EQ(DFPLAYER_EQ_NORMAL); // Устанавливаем эквалайзер с настройками NORMAL
```

Функция outputDevice();

- Назначение: указываем тип устройства, с которого будут воспроизводиться файлы;
- Синтаксис: **outputDevice(ПАРАМЕТР)**;
- Параметры:
    - **ПАРАМЕТР** - один из 5 источников:
        - **DFPLAYER_DEVICE_U_DISK** - подключен USB-диск;
        - **DFPLAYER_DEVICE_SD** - подключена SD-карта;
        - **DFPLAYER_DEVICE_AUX** - модуль используется как внешняя звуковая карта;
        - **DFPLAYER_DEVICE_SLEEP** - модуль в режиме сна;
        - **DFPLAYER_DEVICE_FLASH** - подключена USB-FLASH карта;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD); // Устанавливаем источником SD-карту
```

Функция sleep();

- Назначение: перевод плеера в режим сна;
- Синтаксис: **sleep()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.sleep(); // Установить режим сна
```

Функция reset();

- Назначение: сброс всех настроек;
- Синтаксис: **reset()**;
- Параметры: нет
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.reset(); // Сбросить все настройки
```

Функция enableDAC;

- Назначение: включение ЦАП;
- Синтаксис: **enableDAC()**;
- Параметры: нет
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
```

```
myDFPlayer.enableDAC(); // Включить АЦП
```

Функция disableDAC;

- Назначение: отключение ЦАП;
- Синтаксис: **disableDAC()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.disableDAC(); // Выключить АЦП
```

Функция outputSetting();

- Назначение: настройка усиления выходного сигнала;
- Синтаксис: **outputSetting(ПАРАМЕТР_1, ПАРАМЕТР_2)**;
- Параметры:
  - **ПАРАМЕТР_1** - включить/выключить усиление выходного сигнала (true / false);
  - **ПАРАМЕТР_2** - коэффициент усиления звука (от 0 до 31);
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.outputSetting(true, 15); // Включить усиление с коэффициентом усиления 15
```

Функция next();

- Назначение: смена трека на следующий;
- Синтаксис: **next()**;

- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.next(); // Включить воспроизведение следующего трека
```

Функция previous();

- Назначение: смена трека на предыдущий;
- Синтаксис: **previous()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.previous(); // Включить воспроизведение предыдущего трека
```

Функция play();

- Назначение: включить воспроизведение первого или указанного трека
- Синтаксис: **play(ПАРАМЕТР)**;
- Параметры:
  - **Если ПАРАМЕТР указан** - начать воспроизведение указанного трека в первой папке;
  - **Если ПАРАМЕТР не указан** - начать воспроизведение первого трека в первой папке;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.play(1); // Включить воспроизведение указанного (1) трека
```

Функция loop();

- Назначение: включить непрерывное воспроизведение указанного трека;
- Синтаксис: **loop(ПАРАМЕТР)**;
- Параметры:
  - **ПАРАМЕТР** - номер трека, который следует включать непрерывно;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.loop(1); // Включить непрерывное воспроизведение указанного (1) трека
```

Функция pause();

- Назначение: поставить трек, который воспроизводится, на паузу;
- Синтаксис: **pause()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.pause(); // Поставить воспроизведение на паузу
```

Функция start();

- Назначение: продолжить воспроизводить трек, если до этого он был поставлен на паузу;
- Синтаксис: **start()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;

- Пример:

```
myDFPlayer.start(); // Продолжить воспроизведение трека (после паузы)
```

Функция playFolder();

- Назначение: играть указанный трек в указанной папке;
- Синтаксис: **playFolder(ПАРАМЕТР_1, ПАРАМЕТР_2)**;
- Параметры:
  - **ПАРАМЕТР_1** - номер папки (от 1 до 99);
  - **ПАРАМЕТР_2** - номер трека (от 1 до 255);
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.playFolder(15, 4); // Включить воспроизведение указанного трека из указанной папки
```

Функция enableLoopAll;

- Назначение: включить непрерывное воспроизведение всех треков;
- Синтаксис: **enableLoopAll()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.enableLoopAll(); // Включить непрерывное воспроизведение всех треков
```

Функция disableLoopAll();

- Назначение: выключить непрерывное воспроизведение всех треков;

- Синтаксис: **disableLoopAll()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.disableLoopAll(); // Выключить непрерывное воспроизведение всех треков
```

Функция playMp3Folder();

- Назначение: начать воспроизведение указанного трека в папке "MP3";
- Синтаксис: **playMp3Folder(ПАРАМЕТР)**;
- Параметры:
  - **ПАРАМЕТР** - номер трека в папке "MP3" (от 0 до 65535);
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.playMp3Folder(4); // Включить воспроизведение указанного трека из папки MP3
```

Функция advertise();

- Назначение: включить воспроизведение указанного трека в режиме "ADVERTISE" (реклама)
- Синтаксис: **advertise(ПАРАМЕТР)**;
- Параметры:
  - **ПАРАМЕТР** - номер трека в папке "ADVERT";
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```

```

```
myDFPlayer.advertise(3); // Включить воспроизведение указанного трека из папки ADVERT
```

Функция stopAdvertise();

- Назначение: прекратить воспроизведение треков в режиме "ADVERTISE" (реклама)
- Синтаксис: **stopAdvertise()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.stopAdvertise(); // Выключить воспроизведение трека в режиме ADVERTISE
```

Функция playLargeFolder();

- Назначение: начать воспроизведение указанного трека из указанной папки с бОльшим, чем обычно, количеством треков;
- Синтаксис: **playLargeFolder(ПАРАМЕТР_1, ПАРАМЕТР_2)**;
- Параметры:
  - **ПАРАМЕТР_1** - номер папки (от 1 до 10);
  - **ПАРАМЕТР_2** - номер трека (от 1 до 999);
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.playLargeFolder(2, 999); // Включить воспроизведение указанного трека из указанной папки
```

Функция loopFolder();

- Назначение: включить непрерывное воспроизведение всех файлов в указанной папке;
- Синтаксис: **loopFolder(ПАРАМЕТР)**;
- Параметры:

    - **ПАРАМЕТР** - номер папки (от 1 до 99);
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.loopFolder(5); // Включить непрерывное воспроизведение указанной (5) папки
```

Функция randomAll();

- Назначение: перемешать порядок воспроизведения всех треков;
- Синтаксис: **randomAll()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.randomAll(); // Включить случайное воспроизведение всех треков
```

Функция enableLoop();

- Назначение: включить непрерывное воспроизведение треков;
- Синтаксис: **enableLoop()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.enableLoop(); // Включить непрерывное воспроизведение треков
```

Функция disableLoop();

- Назначение: выключить непрерывное воспроизведение треков;
- Синтаксис: **disableLoop()**;
- Параметры: нет;
- Возвращаемые значения: нет;
- Примечание: нет;
- Пример:

```
myDFPlayer.disableLoop(); // Выключить непрерывное воспроизведение треков
```

Функция readState();

- Назначение: запрос состояния плеера
- Синтаксис: **readState()**;
- Параметры: нет;
- Возвращаемые значения:
  - **512** - воспроизведение остановлено;
  - **513** - воспроизведение выполняется;
  - **514** - воспроизведение стоит на паузе;
- Примечание: нет;
- Пример:

```
myDFPlayer.readState(); // Состояние проигрывания файла
```

Функция readVolume();

- Назначение: запрос уровня громкости;
- Синтаксис: **readVolume()**;
- Параметры: нет;
- Возвращаемые значения:
  - Значение уровня громкости (от 0 до 30)

- Примечание:
- Пример:

```
myDFPlayer.readVolume(); // Значение установленной громкости
```

Функция readEQ();

- Назначение: запрос номера установленного эквалайзера;
- Синтаксис: **readEQ()**;
- Параметры: нет;
- Возвращаемые значения:
  - Значение от 0 до 5, соответствующее следующему списку:
    - **0 - DFPLAYER_EQ_NORMAL;**
    - **1 - DFPLAYER_EQ_POP;**
    - **2 - DFPLAYER_EQ_ROCK;**
    - **3 - DFPLAYER_EQ_JAZZ;**
    - **4 - DFPLAYER_EQ_CLASSIC;**
    - **5 - DFPLAYER_EQ_BASS;**
- Примечание: нет;
- Пример:

```
myDFPlayer.readEQ(); // Тип установленного эквалайзера
```

Функция readFileCounts();

- Назначение: запрос общего количества файлов во всех папках;
- Синтаксис: **readFileCounts()**;
- Параметры: нет;
- Возвращаемые значения:
  - Общее количество файлов;

- Примечание:
- Пример:

```
myDFPlayer.readFileCounts(); // Количество файлов на SD-карте
```

Функция readCurrentFileNumber();

- Назначение: запрос номера трека, который воспроизводится;
- Синтаксис: **readCurrentFileNumber()**;
- Параметры: нет;
- Возвращаемые значения:
  - Номер трека;
- Примечание: нет;
- Пример:

```
myDFPlayer.readCurrentFileNumber(); // Номер проигрываемого трека
```

Функция readFileCountsInFolder();

- Назначение: запрос количества треков в указанной папке;
- Синтаксис: **readFileCountsInFolder()**;
- Параметры: номер папки;
- Возвращаемые значения:
  - Число файлов в указанной папке;
- Примечание:
- Пример:

```
myDFPlayer.readFileCountsInFolder(3); // Количество файлов в указанной (3) папке
```

Функция readFolderCounts();

- Назначение: запрос общего количества папок;
- Синтаксис: **readFolderCounts()**;
- Параметры: нет;
- Возвращаемые значения:
  - Общее количество папок;
- Примечание:
- Пример:

```
myDFPlayer.readFolderCounts(); //
```

Функция readType();

- Назначение: запрос кода ошибки/состояния работы плеера;
- Синтаксис: **readType()**;
- Параметры: нет;
- Возвращаемые значения:
  - Одно из следующих значений:
    - **TimeOut** - время ожидания ответа истекло;
    - **WrongStack** - не верный стек;
    - **DFPlayerCardInserted** - карта установлена;
    - **DFPlayerCardRemoved** - карта изъята;
    - **DFPlayerCardOnline** - карта на связи;
    - **DFPlayerUSBInserted** - USB установлено
    - **DFPlayerUSBRemoved** - USB изъято;
    - **DFPlayerPlayFinished** - проигрывание файла завершено;
      - в связке с функцией **read()** можно так же вернуть номер трека;
    - **DFPlayerError** - наличие ошибки
      - в связке с функцией **read()** можно так же получить дополнительный параметр:
        - **Busy** - карта не найдена;

        - **Sleeping** - плеер в режиме сна;
        - **SerialWrongStack** - ошибка стека;
        - **CheckSumNotMatch** - ошибка контрольной суммы;
        - **FileIndexOut** - номер трека превысил допустимый диапазон;
        - **FileMismatch** - файл не найден;
        - **Advertise** - плеер в режиме "ADVERTISE" (реклама);
- Примечание:
    - Функция используется в связке с функцией **read()** для получения дополнительных параметров данных;
- Пример:

```
myDFPlayer.readType(); // запрос кода ошибки/состояния работы плеера
```

Функция read();

- Назначение: запрос параметра кода ошибки/состояния плеера;
- Синтаксис: **read()**;
- Параметры: нет;
- Возвращаемые значения:
    - Параметр кода ошибки/состояния плеера.
- Примечание:
- функция используется в связке с функцией **readType()** для получения дополнительных параметров данных;
- Пример:

```
myDFPlayer.read(); // запрос параметра кода ошибки/состояния работы плеера;
```

Функция waitAvailable();

- Назначение: ожидание ответа от плеера;
- Синтаксис: **waitAvailable(ПАРАМЕТР)**;
- Параметры:

    - **Если ПАРАМЕТР установлен** - время в мс, в течении которого ожидается ответ;
    - **Если ПАРАМЕТР не установлен** - ожидание ответа равно 0мс;
- Возвращаемые значения: нет;
- Примечание:
    - Необязательная функция, которую можно использовать для дополнительной проверки связи с плеером;
- Пример:

```
myDFPlayer.waitAvailable(); // запрос связи с плеером;
```

Применение:

- звуковое сопровождение проектов, озвучание эффектов;
- режим воспроизведения голосовых сообщений, построение системы охраны и оповещения;