МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Факультет физики и информационных технологий
Кафедра общей физики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

МОБИЛЬНЫЙ ПРОИГРЫВАТЕЛЬ АУДИО ФАЙЛОВ

ГГУ 1-39 03 02 024 КП

Исполнитель:
Студент группы МС-32: _____ Чвалов К. Р.

Научный руководитель:
Старший преподаватель
Кафедры общей физики _____ Подалов М. А.

Гомель 2023

РЕФЕРАТ

МОБИЛЬНЫЙ ПРОИГРЫВАТЕЛЬ АУДИО ФАЙЛОВ: Курсовой проект / К. Р. Чвалов. – Гомель : ГГУ имени Ф. Скорины, 2023, – п.з. – 51 с., чертежей – 2 л. формата А4.

Предметом проектирования является мобильный проигрыватель аудио файлов.

Цель работы: освоить принципы работы проигрывателей аудио файлов, спроектировать простой мобильный проигрыватель аудиофайлов, работающий автономно за счет аккумулятора, с возможностью подзарядки, имеющий кнопку управления, разъем подключения наушников и слот для SD-карты.

В процессе выполнения курсового проекта были **решены следующие задачи**:

- выполнен анализ предметной области;
- подобраны необходимые компоненты для проигрывателя;
- разработана структурная электрическая схема проигрывателя;
- разработана принципиальная электрическая схема проигрывателя;
- разработан алгоритм функционирования проигрывателя;

СОДЕРЖАНИЕ

Введение

В современном мире мобильные устройства являются неотъемлемой частью повседневной жизни почти каждого человека. И среди всех устройств, смартфон, без сомнений, занимает лидирующие позиции, так как он являются, по сути, универсальным мобильным устройством, совмещающим в себе множество функций и возможностей других устройств, в том числе и аудиопроигрывателя.

Однако, не всегда встроенный проигрыватель удовлетворяет потребности всех пользователей, особенно если речь идет о специализированном устройстве для воспроизведения аудиофайлов. В связи с этим возникает необходимость использования мобильных проигрывателей, которые могут удовлетворять потребности пользователей и быть адаптированными к конкретным условиям использования.

Цель: спроектировать простой мобильный проигрыватель аудиофайлов, работающий автономно за счет аккумулятора, с возможностью подзарядки, имеющий кнопки управления, разъем подключения наушников и слот для SD-карты.

Задачи:

- Выполнить анализ предметной области;
- Подобрать необходимые компоненты для аудиопроигрывателя;
- Разработать структурную электрическую схему аудиопроигрывателя;
- Разработать принципиальную электрическую схему аудиопроигрывателя;
- Разработать алгоритм функционирования аудиопроигрывателя;

1 Анализ предметной области

Аудиоплеер представляет собой портативное устройство, предназначенное для хранения и воспроизведения музыкальных файлов, которые хранятся в цифровом виде на карте памяти.

В рамках курсового проекта необходимо разработать мобильный аудиопроигрыватель. Для этого необходимо рассмотреть ряд важных аспектов:

- Форматы аудиофайлов. Существует множество форматов аудиофайлов, каждый из которых имеет свои особенности и назначение. Некоторые из наиболее распространенных форматов: MP3, WAV, AAC, FLAC, OGG.
- Хранение аудиофайлов. Аудиофайлы могут храниться на различных устройствах, например, на жестких дисках, флеш-накопителях, смартфонах, планшетах. Важным аспектом является возможность проигрывателя работать с различными источниками хранения файлов.
- Аппаратная часть. Необходимо определить, какие компоненты необходимы для создания проигрывателя, например, кнопки, аудио-усилитель, платы расширения, микроконтроллер.
- Энергопотребление. Мобильный проигрыватель должен работать автономно, благодаря аккумулятору и иметь достаточно низкое энергопотребление, чтобы обеспечить длительное время автономной работы. При этом проигрыватель должен иметь возможность подзарядки.
- Возможности расширения. При разработке мобильного проигрывателя важно предусмотреть возможность его дальнейшей модернизации и расширения функциональности. Например, это может быть добавление функции записи звука, поддержки новых форматов аудиофайлов, интеграции с облачными сервисами хранения музыки, добавление Bluetooth-модуля, FM-радио и т.д.

1.1 Цифровой звук. Аудиокодек

Перед тем как разобрать форматы аудиофайлов, необходимо разобраться с процессом прохождения и преобразования звука при воспроизведении каких-либо аудиофайлов через динамик или записи какого-либо звука на микрофон.

Цифровой звук представляет собой результат преобразования аналогового сигнала звукового диапазона, полученного, например с микрофона, в цифровой формат аудиофайлов. Один из методов преобразования – это импульсно-кодовая модуляция (ИКМ). Данный метод состоит в представлении последовательности мгновенных значений уровня сигнала, измеряемого аналого-цифровым преобразователем (АЦП) через равные промежутки времени. [3]

Аудиокодек на аппаратном уровне обозначает отдельную микросхему, которая кодирует и декодирует аналоговый звуковой сигнал в цифровой сигнал и наоборот при помощи аналогово-цифрового и цифро-аналогового преобразователей.

1.2 Характеристики форматов аудио файлов

Форматов аудиофайлов огромное количество, и такие форматы как: *MP3*, *WAV*, *AAC*, *FLAC*, *OGG* – это лишь одни из самых распространенных. При этом все форматы имеют свои характеристики и их всех можно разделить на несколько типов.

Однако начать стоит с характеристик. Любая информация на компьютере хранится в дискретной форме, то есть в форме нулей и единиц. При этом сам звук не дискретный и при записи звука с помощью микрофона получается не дискретный сигнал, а аналоговый.

1.2.1 Общая информация о характеристиках

Аналоговый (или континуальный) сигнал описывается непрерывной функцией времени, т.е. имеет непрерывную линию с непрерывным множеством возможных значений (смотреть Рисунок 1.1.1). [2]

Рисунок 1.1.1 – Аналоговый сигнал

Цифровой сигнал — это сигнал, который можно представить как последовательность определенных цифровых значений. В любой момент времени он может принимать только одно определенное конечное значение (смотреть Рисунок 1.1.2). [2]

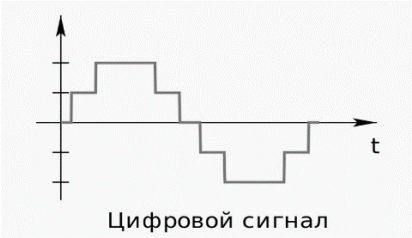


Рисунок 1.1.2 – Цифровой сигнал

Аналоговый сигнал в динамическом диапазоне может принимать любые значения. Для того, чтобы аналоговый сигнал мог быть обработан цифровым

устройством, он должен быть преобразован в цифровой формат. Это осуществляется с помощью двух этапов: дискретизация и квантование. Нет значения, в каком порядке происходят эти процессы.

Дискретизацией называется процесс регистрации (измерения) значения сигнала через определенные промежутки (обычно равные) времени (смотреть Рисунок 1.1.3). [2]

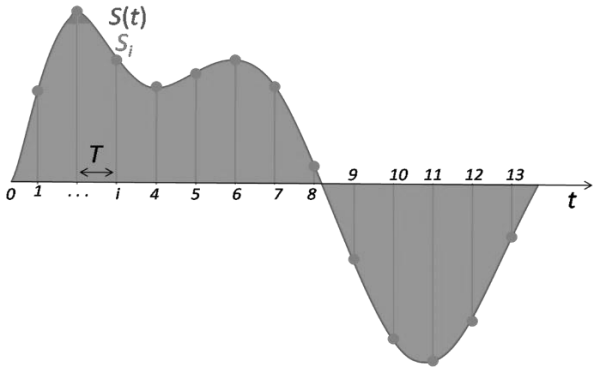


Рисунок 1.1.3 – Дискретизация

Квантование — это процесс разбиения диапазона амплитуды сигнала на определенное количество уровней и округление значений, измеренных во время дискретизации, до ближайшего уровня (смотреть Рисунок 1.1.4). [2]

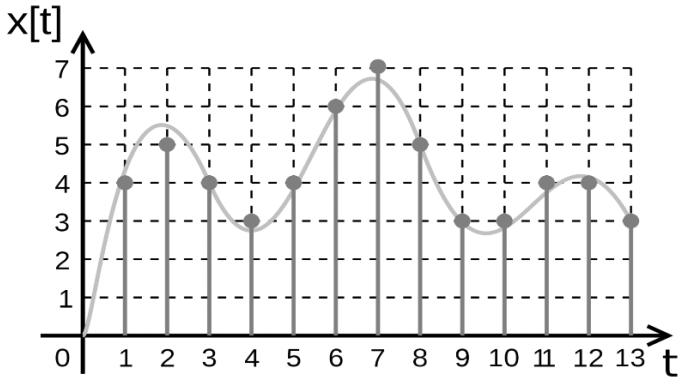


Рисунок 1.1.4 – Квантование

Дискретизация разбивает сигнал по временной составляющей (по вертикали (смотреть Рисунок 1.1.5). [2]

Квантование приводит сигнал к заданным значениям, то есть округляет сигнал до ближайших к нему уровней (смотреть Рисунок 1.1.5). [2]

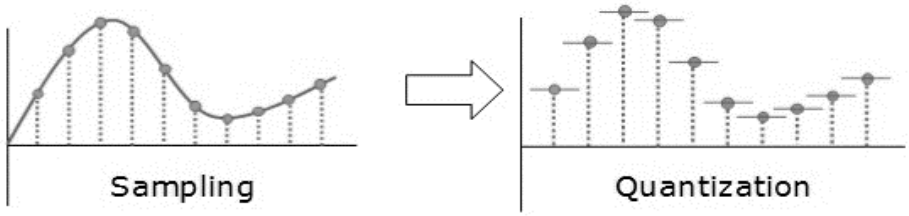


Рисунок 1.1.5 – Дискретизация и квантование

Эти два процесса создают некую координатную систему, которая описывает аудиосигнал определенным значением в любой момент времени. [2]

Цифровым называется сигнал, к которому применены дискретизация и квантование. Оцифровка происходит в аналого-цифровом преобразователе (АЦП). Чем больше число уровней квантования и чем выше частота дискретизации, тем точнее цифровой сигнал соответствует аналоговому (смотреть Рисунок 1.1.6). [2]

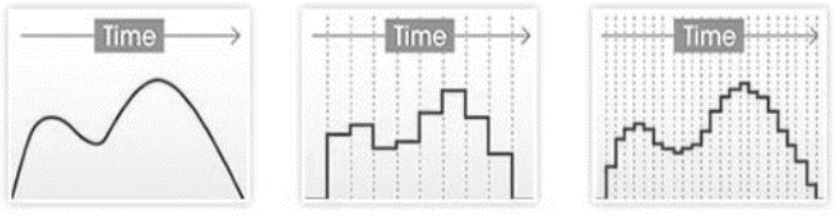


Рисунок 1.1.6 – Уровни квантования и частота дискретизации

1.2.2 Квантование

Уровни квантования нумеруются и каждому уровню присваивается двоичный код (смотреть Рисунок 1.1.7). [2]

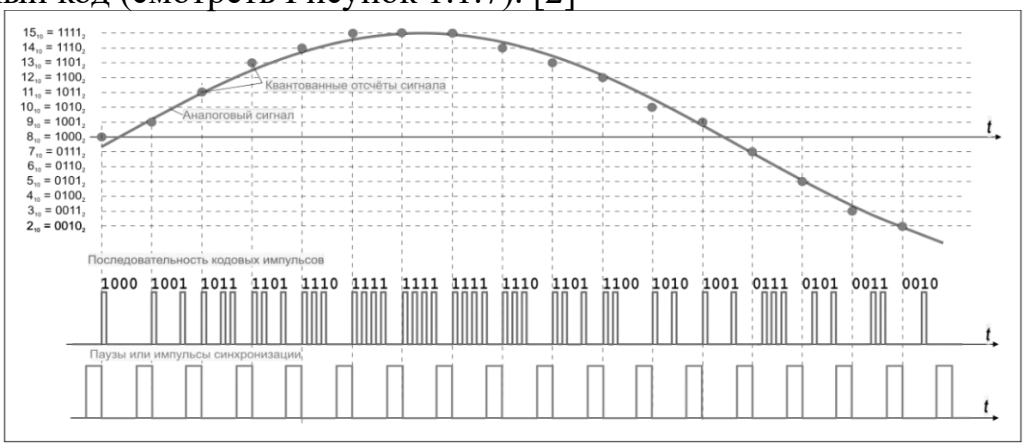


Рисунок 1.1.7 – Описание уровней квантования

Количество битов, которые присваиваются каждому уровню квантования называют *разрядностью* или *глубиной квантования*. Чем выше разрядность, тем больше уровней можно представить двоичным кодом (смотреть Рисунок 1.1.8)**.** [2]

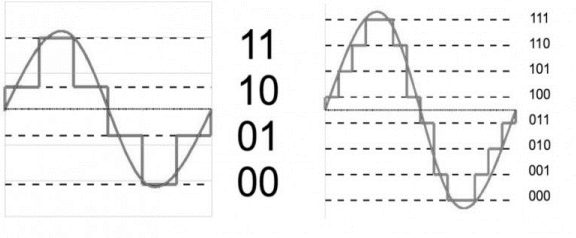


Рисунок 1.1.8 – Глубина квантования

Следующая формула позволяет вычислить число уровней квантования:

$$N = 2^n ,$$

где N – количество уровней квантования,
n – разрядность.

Обычно используют разрядности в 8, 12, 16 и 24 бит. Для формата аудиофайлов Ogg Vorbis используется разрядность до 32 бит. [1]

Следует отметить, что чем ниже разрядность, тем больше округляются значения и тем больше ошибка квантования.

Ошибкой квантования называют отклонение квантованного сигнала от аналогового, т.е. разница между входным значением X и квантованным значением $X'(X - X')$. [2]

1.2.3 Дискретизация

Как уже оговаривалось, *дискретизация* – разбиение сигнала по вертикали и измерение величины значения через определенный промежуток времени. Этот промежуток называется *периодом дискретизации* или *интервалом выборок*. *Частотой выборок*, или *частотой дискретизации* (всеми известный sample rate) называется величина, обратная периоду дискретизации и измеряется в герцах. [2]

Следующая формула позволяет вычислить частоту дискретизации:

$$F = 1/T ,$$

где F – частота дискретизации,
T – период дискретизации.

Обычно используют разрядности 44.1 кГц, 48 кГц, 96 кГц, 192 кГц. [1]

1.2.4 Битрейт

Перед тем, как разобрать такую характеристику, как битрейт, следует описать другую – *число каналов*.

Число каналов в форматах аудиофайлов определяет количество аудио потоков, которые могут быть записаны и воспроизведены в файле. Каждый канал содержит звуковую информацию, которая может быть воспроизведена на определенном динамике или наушниках.

Наиболее используемое количество каналов – это 2 (стерео). Это означает, что звуковая информация разделяется на две части и записывается на два отдельных канала. При прослушивании записи на стерео устройстве звук воспроизводится одновременно из двух каналов, создавая впечатление объемного звука. Также может использоваться 1 канал (моно). В отличие от стерео, где звуковые данные разделены между левым и правым каналами, моно запись содержит только один поток аудио информации.

Теперь о битрейте. *Битрейт* – это характеристика, описывающая объем данных, передаваемых в единицу времени. Битрейт обычно измеряют в битах в секунду (кбит/с или bps). Битрейт может быть переменным, постоянным или усреднённым. [2]

Следующая формула позволяет вычислить битрейт:

$$B = F * N * k ,$$

где B – битрейт,
F – частота дискретизации,
N – разрядность,
k – число каналов.

При постоянном битрейте (constant bitrate, CBR) передача объема потока данных в единицу времени не изменяется на протяжении всей передачи. Главное преимущество — возможность довольно точно предсказать размер конечного файла. Из минусов — не оптимальное соотношение размер/качество, так как «плотность» аудиоматериала в течении музыкального произведения динамично изменяется. [2]

При кодировании переменным битрейтом (VBR), кодек выбирает битрейт исходя из задаваемого желаемого качества. Как видно из названия, битрейт варьируется в течение кодируемого аудиофайла. Данный метод даёт наилучшее соотношение качество/размер выходного файла. Из минусов: точный размер конечного файла очень плохо предсказуем. [2]

Усреднённый битрейт (ABR) является частным случаем VBR и занимает промежуточное место между постоянным и переменным битрейтом. Конкретный битрейт задаётся пользователем. Программа все же варьирует его в определенном диапазоне, но не выходит за заданную среднюю величину. [2]

1.3 Типы форматов аудиофайлов

Форматов аудиофайлов огромное множество. У них есть свои характеристики:

- Разрядность (глубина квантования);
- Частота дискретизации;
- Битрейт;
- Число каналов.

Кроме того, все форматы подразделяются на три типа [1]:

- Форматы аудиофайлов без сжатия (WAV, AIFF);
- Форматы аудиофайлов со сжатием с потерями (APE, FLAC);
- Форматы аудиофайлов со сжатием без потерь(MP3, Ogg);

1.4 Обзор популярных форматов аудио файлов

1.4.1 Форматы аудио файлов без потерь

Здесь будет представлена информация о форматах и без сжатия, и со сжатием, но без потерь.

С целью максимально избежать снижения качества звука во время сжатия аудиофайла разработаны специальные способы сохранения звуковой информации, избегая потерь, которые фактически можно сравнить с архивированием, когда информация просто упаковываются в файл zip, размер которого заметно меньше чем исходные данные. Впоследствии эти данные можно чётко восстановить до каждого бита. Причём сам битрейт для этих файлов неважен. Такие аудиофайлы имеют общее название Lossless – музыка «как есть». Подобные алгоритмы позволяют сжимать файлы в два-три раза. В итоге размер выходит достаточно большой, но при этом с сохранностью исходного звука. [4]

Формат *FLAC* – обеспечивает полную сохранность всех данных из аудиопотока, способен сжимать от 1.4 до 4 раз с битрейтом 350-1010 кбит/с, применяется для создания аудио-коллекций и используются для прослушивания на аппаратуре премиального уровня. [4]

Формат *WAV* – один из старых форматов, который был создан Microsoft совместно с IBM. Это лучший формат аудио для обработки и хранения несжатых аудиоданных, которые по качеству соответствуют CD-дискам. Одна минута звука в нём «весит» порядка 10 Мб. Именно поэтому хранить в .wav фонотеку или пересылать эти аудиофайлы по интернету нецелесообразно. [4]

Формат *APE* – довольно популярный формат, который выпускается для ОС Windows, но при этом имеет несколько неофициальных кодеков для других платформ. Формат поддерживает 8-, 16 и 24-разрядные аудиофайлы, поэтому нашёл широкое применение в профессиональной сфере.для хранения сжатой информации без потерь на устройствах Apple. В отношении степени сжатия он несколько уступает бесплатному FLAC. [4]

1.4.2 Форматы аудио файлов с потерями

Формат *MP3* – один из самых распространённых форматов, который сегодня применяется в файлообменных сетях. Разница между MP3 и FLAC или WAV принципиальная. Низкое качество звука ощущается сразу. Всё дело в битрейте, который во flac может доходить до 1010 кбит/с, тогда как в mp3 составляет в среднем всего 128 кбит/с. В этом и заключается отличие flac от mp3. [4]

Формат *OGG* – открытый формат-контейнер, хорошо зарекомендовавший себя при передаче музыки и речи, как на малых, так и на больших битрейтах. Как и MP3, имеет низкое качество звучания. [4]

1.5 Аппаратная часть проигрывателя аудио файлов

Аппаратную часть мобильного проигрывателя аудиофайлов составляет ряд следующих основных компонентов:

- *Микропроцессор*: отвечает за выполнение всех операций, связанных с проигрыванием аудиофайлов, включая декодирование и обработку аудиоданных. Для разработки устройства имеется множество платформ для разработки: Arduino, ESP, Raspberry Pi и т.д.
- *Память*: хранит аудиофайлы и другие данные, необходимые для проигрывания. Обычно проигрыватели имеют встроенную память, но могут поддерживать и внешние карты памяти.
- *Кодек*: используется для декодирования аудиофайлов, чтобы они могли быть воспроизведены. В зависимости от проигрывателя, он может поддерживать различные форматы аудиофайлов, такие как MP3, AAC, WAV и др.
- *ЦАП*: преобразует цифровые аудиоданные обратно в аналоговый звук, который может быть воспроизведен через наушники или динамики.
- *Усилитель*: усиливает аналоговый звук до необходимого уровня громкости для наушников или динамиков.
- *Аккумулятор*: обеспечивает питание проигрывателя, обычно с помощью литий-ионной батареи.
- *Кнопки управления*: используются для управления проигрыванием, перемоткой и регулировкой громкости.
- *Разъемы*: для подключения наушников или динамиков, а также для зарядки и подключения к компьютеру.

Это основные компоненты мобильного проигрывателя аудиофайлов, которые обычно встречаются в большинстве моделей. Однако, некоторые модели могут иметь дополнительные функции и компоненты, такие как радио, Bluetooth, Wi-Fi, GPS и т.д.

1.5.1 Аппаратная платформа Arduino

Arduino – это электронная платформа, которая состоит из аппаратной и программной частей. Аппаратная часть включает в себя микроконтроллер, периферийные устройства и различные модули, а программная - язык программирования и среду разработки. [5]

Язык программирования Arduino основан на языке C++, но упрощен и адаптирован для программирования микроконтроллеров. Язык программирования Arduino позволяет создавать программы, использующие различные периферийные устройства и модули, такие как сенсоры, моторы, светодиоды и другие устройства. [5]

Загрузка программы в микроконтроллер происходит через USB-порт. Для этого необходимо подключить плату Arduino к компьютеру и загрузить скетч (программу) с помощью среды разработки Arduino. Среда разработки Arduino позволяет проверять и отлаживать программы, а также загружать их в микроконтроллер. [5]

Классический конструктив Arduino представляет собой плату, которая содержит микроконтроллер AVR от компании Atmel, различные периферийные устройства и разъемы для подключения дополнительных модулей. Изображение Arduino UNO (смотреть Рисунок 1.5.1).

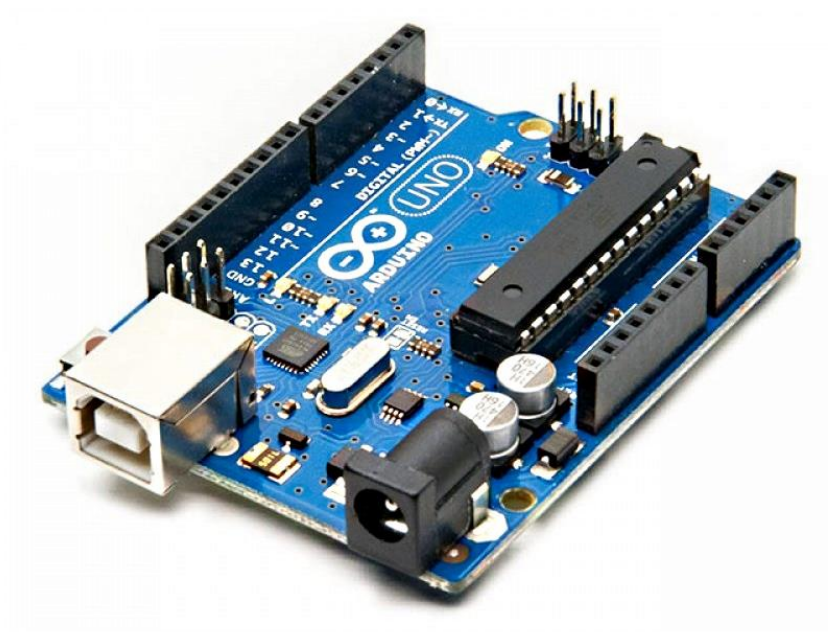


Рисунок 1.5.1 – Arduino UNO

Миниатюрный конструктив Arduino Mini и Arduino Nano являются компактными версиями платы Arduino. Они содержат микроконтроллер и некоторые периферийные устройства, но не имеют разъемов для подключения дополнительных модулей. Изображение Arduino NANO (смотреть Рисунок 1.5.2).

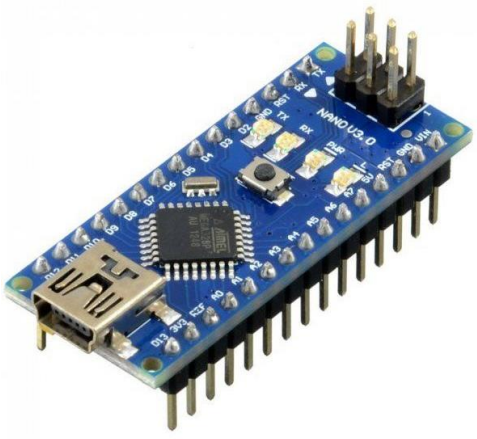


Рисунок 1.5.2 – Arduino NANO

В последнее время наряду с AVR-микроконтроллерами платформа Arduino начала использовать микроконтроллеры семейства ARM и процессоры ESP8266 и ESP32. Микроконтроллеры ARM имеют более высокую производительность и больше памяти, чем AVR, что позволяет создавать более сложные программы. Процессоры ESP8266 и ESP32 имеют встроенный Wi-Fi модуль, что делает их подходящими для создания устройств Интернета вещей. [5]

Периферийные устройства, доступные на платформе Arduino, включают в себя различные сенсоры, дисплеи, моторы, светодиоды и другие устройства. Они подключаются к плате Arduino через разъемы, что позволяет легко и быстро расширять функциональность устройства. [5]

В целом, платформа Arduino является удобной и доступной платформой для разработки электронных устройств, которая позволяет создавать прототипы и проекты без глубоких знаний электроники и программирования. Arduino подходит как для начинающих, так и для профессиональных разработчиков. [5]

Одним из основных преимуществ Arduino является наличие большого сообщества разработчиков, которые создают библиотеки и проекты для платформы Arduino. Благодаря этому, можно быстро и легко найти решение для большинства задач. [5]

Также стоит отметить, что Arduino имеет открытый исходный код, что позволяет пользователям свободно распространять и изменять его код. Это делает платформу Arduino гибкой и универсальной для различных проектов.

В заключение можно сказать, что платформа Arduino позволяет быстро создавать прототипы и проекты, а также является удобным инструментом для изучения электроники и программирования. Она имеет множество периферийных устройств и поддерживает различные микроконтроллеры, что позволяет выбирать наиболее подходящую конфигурацию для каждого проекта.

1.5.2 Семейство плат ESP

Семейство плат ESP включает в себя несколько моделей, основанных на микроконтроллерах производства Espressif Systems. Наиболее популярными моделями являются ESP8266 и ESP32. [6]

Программирование плат ESP производится на языке программирования C++, используя среду разработки Arduino IDE. В Arduino IDE предусмотрено множество библиотек и примеров кода, что значительно упрощает процесс программирования. [6]

Для загрузки программы на плату ESP используется USB-кабель и встроенный в микроконтроллер загрузчик. Для этого не требуется использование дополнительных программаторов. [6]

Платы ESP имеют множество периферийных устройств, таких как порты ввода-вывода, аналоговые входы, интерфейсы UART, SPI, I2C, Wi-Fi и

Bluetooth. Также на платах ESP установлено большое количество внутренней памяти и возможность подключения внешних модулей памяти. [6]

Классический конструктив платы ESP представляет собой небольшую плату с микроконтроллером, периферийными устройствами и разъемами для подключения. Однако, для удобства использования и компактности, существуют также миниатюрные конструктивы плат ESP. [6]

Микроконтроллеры, используемые на платах ESP, являются высокопроизводительными и имеют большое количество встроенных функций, таких как генераторы случайных чисел, криптографические функции и другие. Наиболее распространенными микроконтроллерами для плат ESP являются чипы из семейства ESP8266 и ESP32. Изображение ESP8266 (смотреть Рисунок 1.5.3). [6]

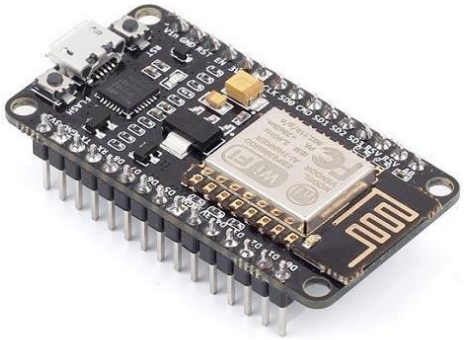


Рисунок 1.5.3 – ESP8266

ESP8266 является более старой и бюджетной моделью, которая имеет ограниченные возможности по сравнению с ESP32. ESP32 имеет больше периферийных устройств, встроенный Wi-Fi и Bluetooth, более высокую скорость работы и большую внутреннюю память. Изображение ESP32 (смотреть Рисунок 1.5.4). [6]

Рисунок 1.5.4 – ESP32

В целом, семейство плат ESP является универсальным и удобным инструментом для создания различных проектов, включая Интернет вещей (IoT), системы управления домашними устройствами, робототехнику и другие.

1.5.3 Микрокомпьютеры Raspberry Pi

Микрокомпьютеры Raspberry Pi – это набор из одноплатных компьютеров, созданных в Великобритании фондом Raspberry Pi Foundation. Эти компьютеры созданы для образовательных целей и разработки проектов в области электроники и программирования. [7]

Существует несколько моделей Raspberry Pi, каждая из которых имеет свои особенности и технические характеристики.

Raspberry Pi Zero – это наименьшая модель, которая имеет размеры 65x30 мм и весит всего 9 граммов. Она оснащена процессором Broadcom BCM2835 с тактовой частотой 1 ГГц, 512 МБ оперативной памяти, а также портами mini-HDMI, USB и microSD. [7]

Raspberry Pi 4 – это самая мощная модель, оснащенная 4-ядерным процессором ARM Cortex-A72 с тактовой частотой 1,5 ГГц, 2 или 4 ГБ оперативной памяти, портами Gigabit Ethernet, USB 3.0, HDMI, а также поддерживает беспроводные интерфейсы Wi-Fi и Bluetooth. Изображение Raspberry Pi 4 (смотреть Рисунок 1.5.5). [7]

Рисунок 1.5.4 – Raspberry Pi 4

Кроме того, существуют модели Raspberry Pi 2, Raspberry Pi 3 и Raspberry Pi Zero W, которые также имеют свои особенности. [7]

Программирование Raspberry Pi производится на языке Python, а также на C/C++, Ruby и других языках программирования. Для управления платой используется операционная система Raspbian, основанная на Linux. [7]

Raspberry Pi имеет множество периферийных устройств, включая порты GPIO, Ethernet, USB, HDMI, аудио и видео, а также Wi-Fi и Bluetooth. Платы также могут быть расширены с помощью дополнительных модулей, таких как камеры, дисплеи, датчики и другие. [7]

Микрокомпьютеры Raspberry Pi можно использовать для различных проектов, включая создание серверов, мультимедийных систем, управления домашними устройствами, робототехники и многого другого. В целом, Raspberry Pi является удобным и доступным инструментом для обучения программированию и разработке электронных устройств.

1.5.4 Сравнение трёх представленных платформ

Arduino, *ESP* и *Raspberry Pi* - это разные устройства, которые предназначены для разных целей.

Arduino – это платформа быстрой разработки электронных устройств, которая подходит для начинающих и профессионалов. Arduino можно использовать для создания простых устройств. Он имеет ограниченные вычислительные возможности, поэтому он не подходит для сложных проектов, таких как создание мультимедийных систем или управления домашними устройствами.

ESP – это набор одноплатных компьютеров, который предназначен для создания проектов в области интернета вещей (IoT). Он имеет более высокую вычислительную мощность, чем Arduino, и поддерживает беспроводные интерфейсы, такие как Wi-Fi и Bluetooth, что делает его удобным для создания устройств, которые нужно удаленно управлять или мониторить.

Raspberry Pi – это полноценный компьютер на одной плате, который имеет высокую вычислительную мощность и полноценную операционную систему, такую как Raspbian. Raspberry Pi подходит для создания различных проектов, включая мультимедийные системы, серверы, управление домашними устройствами и т.д. . Raspberry Pi имеет большое количество периферийных устройств и расширений, что делает его универсальным инструментом для разработки и создания различных проектов.

Таким образом, Arduino, ESP и Raspberry Pi предназначены для разных целей и имеют разные характеристики, которые должны быть учтены при выборе устройства для конкретного проекта.

Таблица 1 – Сравнение трех платформ, их плюсы и минусы

Характеристика	Arduino	ESP	Raspberry Pi
Простота разработки	+	-	-
Вычислительная мощность	-	+	++
Скорость разработки	++	+	-
Сообщество	++	+	++
Цена	++	+	--
Назначение	Электронные устройства, микроконтроллеры	Интернет вещей, беспроводные интерфейсы	Полноценный компьютер на одной плате
Примеры проектов	Датчики, светодиодные ленты, проигрыватели и т.д.	Устройства для удаленного управления или мониторинга, IoT-проекты	Мультимедийные системы, серверы, управление домашними устройствами

1.6 Аудио усилители, их классификация и сравнение

Усилитель мощности используется для управления громкоговорителями и должен обеспечивать передачу сигнала с минимальными искажениями при подаче высокого напряжения и большого тока на нагрузку с низким сопротивлением в широком диапазоне частот. [8]

Эффективность работы усилителя является важным показателем, однако также важно учитывать степень генерации тепла и энергопотребление устройства. Усилитель может быть реализован как самостоятельное устройство или быть включенным в состав другого устройства. [8]

Усилитель мощности состоит из каскадов усиления, которые могут быть однокаскадными или последовательностью связанных между собой усилительных элементов, цепей нагрузки и связей с другими ступенями. Усилительные элементы могут быть разными, включая электронные лампы, транзисторы или туннельные диоды. [8]

Отрицательные обратные связи используются для стабилизации работы усилителя и снижения искажения сигнала. Различные элементы, такие как термисторы или частотно-зависимые компоненты, могут быть использованы для этой цели. [8]

Аттенюаторы, потенциометры и фильтры могут быть использованы для регулирования степени усиления и частотной характеристики сигнала в разных частях усилителя.

Усилители могут быть *аналоговыми* или *цифровыми*. В аналоговых устройствах входной сигнал усиливается аналоговыми каскадами и выходной сигнал остается аналоговым. В цифровых усилителях входной сигнал после аналогового усиления подвергается аналого-цифровым преобразованиям, а выходной сигнал становится цифровым. [8]

Класс усилителя – это важная характеристика, на которую обращают внимание в первую очередь, она указывает на режим работы усилительного элемента. Класс усилителя обозначается латинской буквой. [8]

1.6.1 Усилитель класса А

Класс А – это самый простой класс усилителей, который использует один проводящий транзистор и почти исключает нелинейные искажения сигнала. Хотя класс А редко применяется в выходных каскадах усилителя мощности из-за низкой эффективности и больших потерь энергии, он хорошо подходит для входных каскадов. Усилители класса А часто используют пары дополнительных транзисторов для управления громкоговорителем на полной мощности. Однако работа усилителей класса А связана с большим выделением тепла, что требует громоздких систем охлаждения и высокого потребления электроэнергии. [8]

Несмотря на низкий КПД (эффективность), усилители класса А считаются золотым стандартом качества среди аудиофилов, так как они

обеспечивают точное и почти неискаженное воспроизведение сигнала. Однако большинство пользователей не готовы мириться с низкой эффективностью, повышенным выделением тепла, малой мощностью и высоким потреблением электроэнергии.

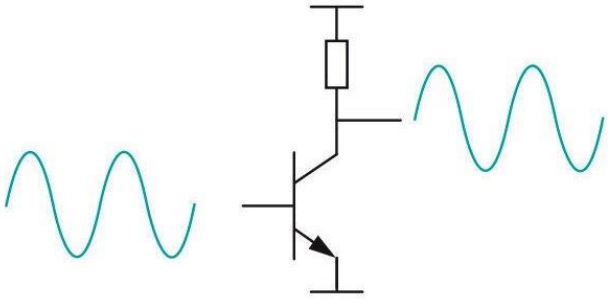


Рисунок 1.6.1 – Работа усилителя класса А

1.6.2 Усилитель класса В

Усилители класса В являются двухтактными и позволяют каждому выходному транзистору проводить только половину волны. По сравнению с классом А, они имеют более высокую эффективность, но в то же время страдают от кроссоверных искажений, которые вносят существенные искажения в сигнал и могут быть восприняты слушателем как неприятные шумы. Именно по этой причине, чистые усилители класса В уже не используются, так как были полностью заменены усилителями класса D. [8]

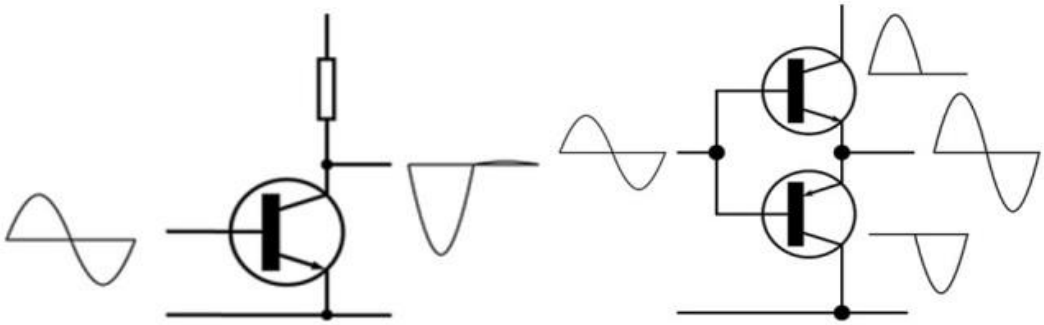


Рисунок 1.6.2 – Работа усилителя класса В

1.6.3 Усилитель класса АВ

Усилители класса АВ являются наиболее распространенным типом усилителей мощности, который широко используется в AV-ресиверах для домашних кинотеатров и стереоусилителях. Этот класс усилителей объединяет преимущества усилителей классов А и В, что позволяет достичь более высокой эффективности по сравнению с усилителями класса А и более низких искажений по сравнению с усилителями класса В. Кроме того, усилители класса АВ имеют более низкую температуру нагрева и простую схемотехнику,

которая не требует высококачественных компонентов. Однако, искажения в усилителях класса АВ все же выше, чем в усилителях класса А. [8]

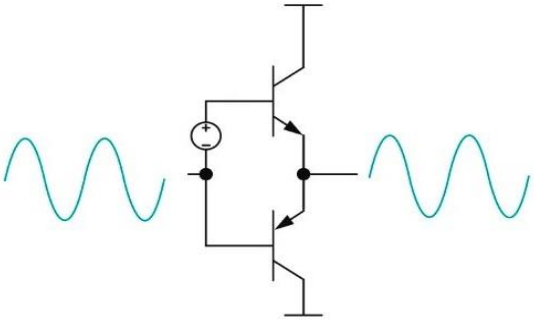


Рисунок 1.6.3 – Работа усилителя класса В

1.6.4 Усилитель класса D

Усилители класса D являются развивающимся и часто используемым типом усилителей, благодаря их высокой эффективности и отсутствию фонового шума. Они имеют компактные размеры и малую теплоотдачу, а также не требуют прогрева перед работой, что делает их популярными в домашней аудиоаппаратуре, профессиональных и портативных устройствах.

Однако, из-за сложности схемотехники и использования качественных компонентов, усилители класса D могут иметь высокую стоимость. Кроме того, качество таких усилителей зависит от типа и частоты модуляции. [8]

В усилителях класса D малой мощности используется одна микросхема, которая не требует дополнительных фильтров, а обратная связь компенсирует искажения сигнала и пульсации питания. Высокая частота модуляции помогает фильтровать "лишние" высокие частоты. [8]

Для профессионального использования в усилителях класса D могут быть встроены конвертеры PCM в DSD, а также цифровой сигнальный процессор (DSP), который компенсирует искажения, вызванные динамиком и характеристиками помещения, обеспечивая максимально качественный звук.

Таким образом, усилители класса D имеют ряд преимуществ, таких как высокая эффективность, компактность и отсутствие фонового шума, но могут иметь высокую стоимость и сложную схемотехнику. Они могут быть особенно полезны для профессионального использования и в портативной технике.

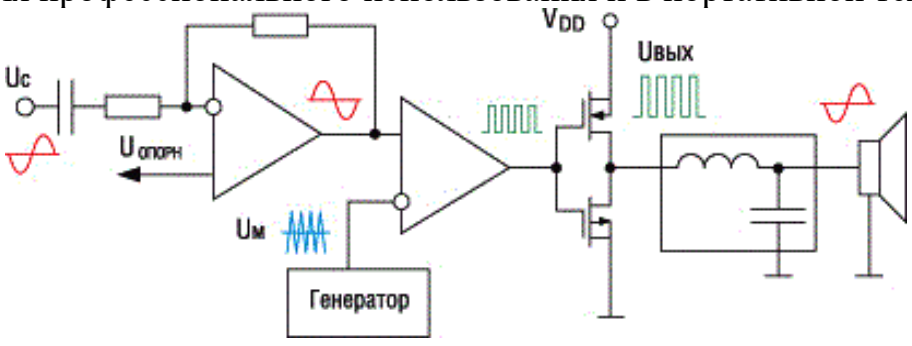


Рисунок 1.6.4 – Работа усилителя класса D

1.6.5 Усилители класса G, H, E, F

Усилители классов G и H являются модификациями класса AB и имеют дополнительные источники напряжения, которые позволяют изменять напряжение питания выходного каскада в зависимости от уровня сигнала, что улучшает КПД. Класс H хорошо подходит для компактных усилителей высокой мощности, которые используются в профессиональных акустических системах. [8]

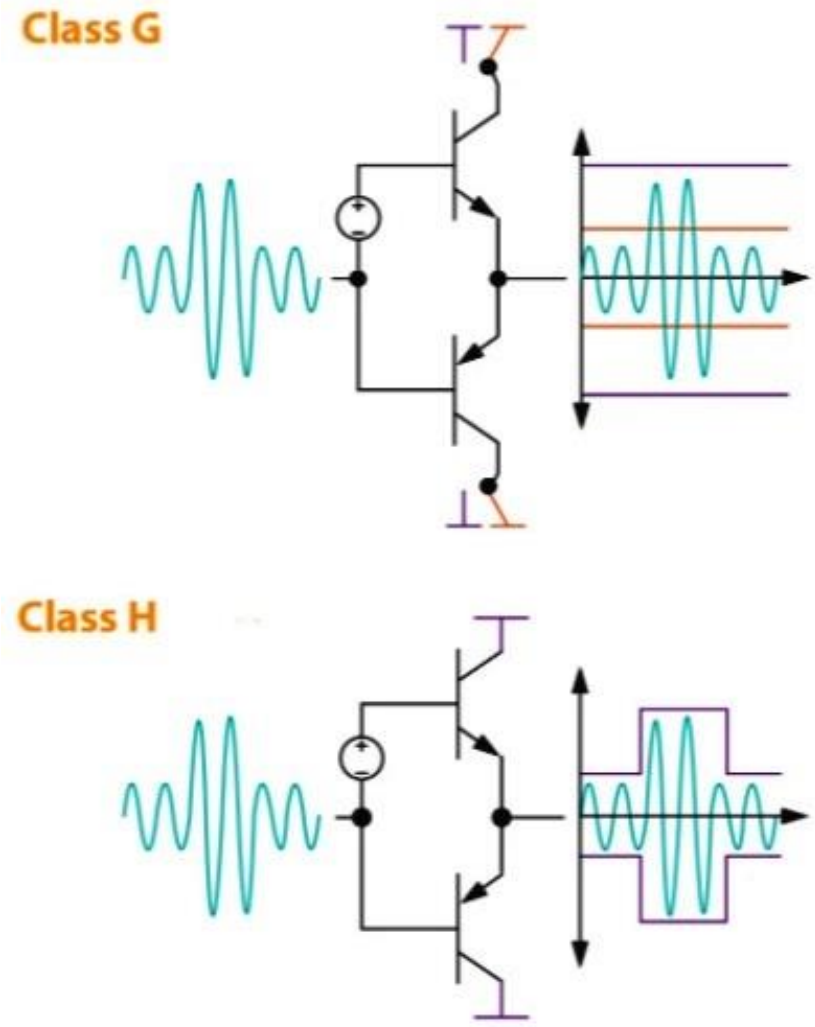


Рисунок 1.6.5 – Работа усилителей класса G и H

Усилители классов E и F обладают еще более высоким КПД, но их использование ограничено из-за больших искажений. Преимуществами этих усилителей являются высокая эффективность, низкая теплоотдача и компактность. Однако, их стоимость может быть высокой. [8]

В целом, усилители классов G, H, E и F представляют различные модификации классического AB-усилителя, которые имеют свои преимущества и недостатки, в зависимости от конкретных потребностей и задач.

1.6.6 Выводы и сравнение классов аудио усилителей

В заключение, можно сказать, что качество усилителя не определяется его классом, а зависит от выбранной схемотехники. Хотя каждый класс имеет свои преимущества и недостатки, КПД является одним из важных показателей, и наиболее высокий он достигается в усилителях класса D. [8]

Таблица 2 – Сравнение классов аудиоусилителей, их плюсы и минусы

Класс усилителя	КПД, %	Достоинства	Недостатки
A	15-35	- минимальные искажения сигнала - высокая точность воспроизведения	- низкая эффективность - высокое потребление энергии - повышенное выделение тепла - малая мощность
B	70	- Высокий КПД	- искажения сигнала
AB	50-70	- КПД выше, чем в усилителях класса A - искажения меньше, чем в усилителях класса B - меньший нагрев, чем в усилителях класса A - простая схемотехника	- недостаточно высокий КПД
D	90	- высокая эффективность - компактные размеры и вес - отсутствие фонового шума - низкая теплоотдача - готовность к работе сразу же после включения	- зависимость качества звука от нагрузки на динамик - сложная схемотехника - цена
H, G	50-70	- высокая эффективность - низкая теплоотдача - компактность	- цена

2 Разработка электрических схем проигрывателя аудио файлов

2.1 Разработка структурной схемы проигрывателя аудио файлов

2.1.1 Обоснование базовых блоков структурной электрической схемы проигрывателя аудио файлов

В рамках курсового проекта для разработки мобильного проигрывателя аудиофайлов была за основу была взята платформа **Arduino Nano** с микроконтроллером ATmega328P.

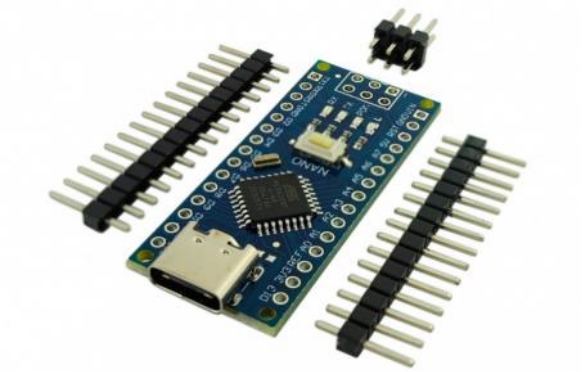


Рисунок 2.1.1 – Arduino Nano V3

Arduino Nano — это уменьшенный аналог Arduino Uno, отличается формфактором платы, которая в 2-2.5 раза меньше (19 х 43 мм), чем Arduino Uno (53 х 69 мм), в отсутствии силового разъема постоянного тока и работе через кабель Mini-B USB. Платформа Nano имеет контакты в виде пинов, поэтому ее легко устанавливать на макетную плату.

На плате используется чип FTDI FT232RL для USB-Serial преобразования и применяется mini-USB кабель для связи с ардуино вместо стандартного. Связь с различными устройствами обеспечивают UART, I2C и SPI интерфейсы. Более подробные параметры представлены в таблице 3.

Таблица 3 – Характеристики Arduino Nano V3

Характеристики	Недостатки
Микроконтроллер	ATmega328P
FLASH	32 KB
EEPROM	1 KB
SRAM	2 KB

Далее для считывания данных с SD карты будет использован специальный **модуль HW-125 для чтения/записи данных с SD карты**, работающий по SPI интерфейсу.

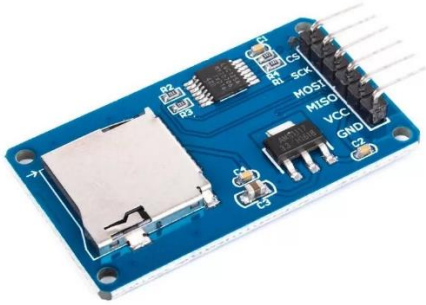


Рисунок 2.1.2 – Модуль HW-125

Данный модуль поддерживает карты Micro SD (до 2 GB) и Micro SDHC (до 32 GB). Модуль питается от напряжение 5В и имеет встроенный регулятор напряжения до 3,3В для SD-карт.

Для усиления звукового сигнала будет использоваться специальный компактный **модуль усилителя звука на чипе LM386**. Микросхема LM386 – это универсальная интегральная микросхема усилителя звука класса АВ. [9]

Таблица 4 – Характеристики модуля усилителя на LM386

Характеристики	Недостатки
Операционный усилитель	LM386M-1
Выходная мощность	ном. 325 мВт, макс. 2 Вт
Сопротивление динамика	8 Ом
Аудиоканалы	1, монофонический
Коэффициент усиления	46 дБ, фиксированный (x200)
Размеры	41 x 13 мм

Для обеспечения автономной работы, необходимо использовать аккумулятор. В рамках курсового проекта будет использоваться литий-ионная **аккумулятор типа 18650 энергоемкостью 3000 мА*ч**.

Рисунок 2.1.3 – Аккумулятор

Для того, чтобы обеспечить зарядку аккумулятора будет использоваться **модуль зарядки с защитой на TP4056**. Модуль основан на чипе TP4056 — контроллере зарядки Li-Ion и Li-Pon аккумуляторов на 3.7В со встроенным термодатчиком.

Рисунок 2.1.4 – Модуль зарядки аккумулятора с защитой на TP4056

Кроме контроллера зарядки TP4056 на плате добавлены два чипа DW01 (схема защиты) и ML8205A (сдвоенный ключ MOSFET), служат для зашиты аккумулятора от переразряда, перезаряда, перегрузки и короткого замыкания.

Таблица 5 – Характеристики модуля зарядки на TP4056

Характеристики	Недостатки
Используемый контроллер	TP4056 и DW01 для зашиты аккумулятора от переразряда и перезаряда
Ток зарядки	до 1А
Входное напряжение	4.5 — 5.5В
Выходное напряжение	От аккумулятора (3,7 В)
Защита от перезаряда	Есть
Защита от переразряда	Есть

Поскольку выходное напряжение от модуля зарядки около 3,7 В, а для Arduino требуется 7-12 В на порт VIN, требуется повышающий DC-DC преобразователь. В рамках курсового проекта будет использоваться **повышающий DC-DC преобразователь HW-045** на основе чипа MT3608.

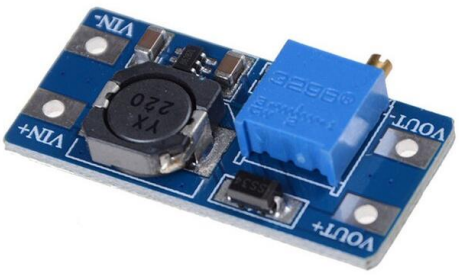


Рисунок 2.1.5 – Повышающий DC-DC модуль HW-045 на чипе MT3608

Таблица 6 – Характеристики повышающего DC-DC модуля на MT3608

Характеристики	Недостатки
Максимальный выходной ток	2A
Входное напряжение	2 – 24 В
Максимальное выходное напряжение	28 В
КПД	< 94%

2.2 Разработка принципиальной схемы проигрывателя аудио файлов

2.2.1 Обоснование выбора САПР для разработки принципиальной электрической схемы проигрывателя аудио файлов

САПР (система автоматизированного проектирования) - это комплекс программных средств, используемый для автоматизации процессов проектирования и создания различных изделий. САПР позволяет значительно ускорить и упростить процесс проектирования, повысить качество разрабатываемых изделий и сократить время и затраты на проектирование.

САПР используются в различных отраслях, таких как машиностроение, электроника, автомобильная и авиационная промышленности, строительство и многие другие. С помощью САПР можно проектировать и моделировать различные объекты, такие как механизмы, электрические схемы, печатные платы, здания, дороги и т.д.

В рамках курсового проекта САПР необходима для проектирования принципиальной электрической схемы мобильного проигрывателя аудио файлов. Ниже будут перечислены наиболее популярные САПР для разработки принципиальных электрических схем, печатных плат и т.д.

Altium Designer – это мощная САПР, предназначенная для проектирования электронных схем и печатных плат. Она объединяет в себе все необходимые инструменты для проектирования, от создания схемы до генерации готового к производству проекта.

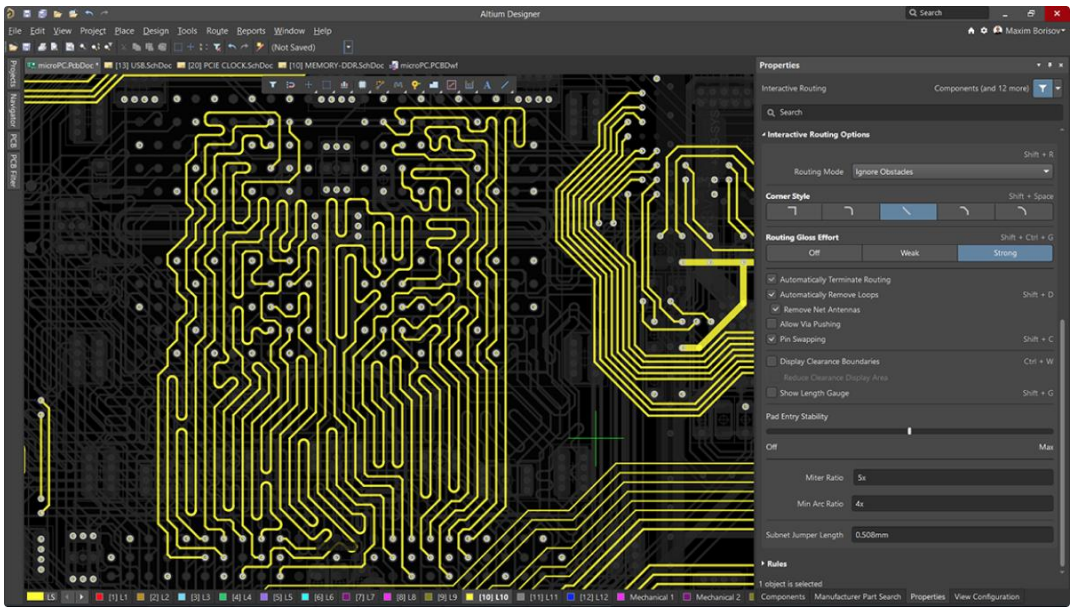


Рисунок 2.2.1 – Интерфейс САПР Altium Designer

Плюсы Altium Designer:

- Богатый набор инструментов для создания и редактирования электронных схем и печатных плат.
- Интуитивно понятный пользовательский интерфейс, который делает процесс проектирования более простым и удобным.
- Высокая точность моделирования и анализа с помощью встроенных инструментов симуляции и анализа.
- Поддержка многослойных плат и гибкая система маршрутизации.
- Мощный набор инструментов для создания 3D-моделей и обеспечения взаимодействия между компонентами.
- Возможность работы с библиотеками компонентов и импорта компонентов из других источников.
- Высокий уровень автоматизации процессов проектирования, что позволяет значительно сократить время и затраты на проектирование.

Минусы Altium Designer:

- Высокая стоимость, что может быть проблемой для малых и средних компаний.
- Высокий порог входа для новичков, из-за множества функций и настроек, которые нужно освоить.
- Ограниченная интеграция с некоторыми другими инструментами и процессами проектирования.

KiCAD – это бесплатная и открытая САПР для проектирования электронных схем и печатных плат. Она предоставляет все необходимые инструменты для создания проектов любой сложности и позволяет легко экспортировать данные во многие форматы.

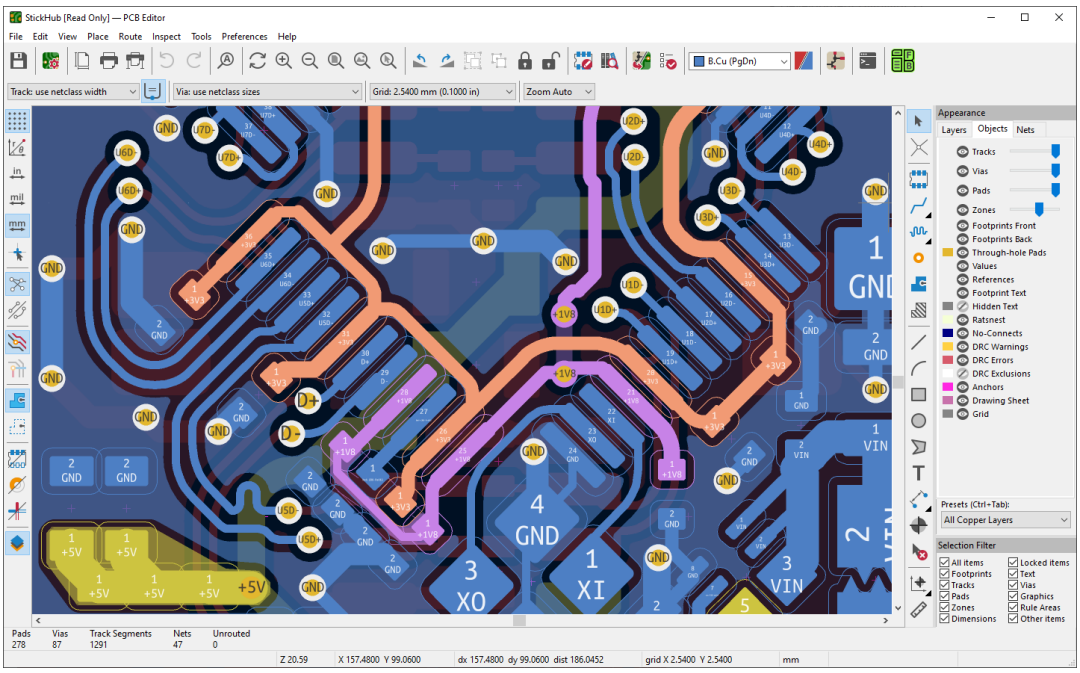


Рисунок 2.2.2 – Интерфейс САПР KiCAD

Плюсы KiCAD:

- Бесплатная и открытая система, доступная для всех.
- Интуитивно понятный пользовательский интерфейс, который делает процесс проектирования более простым и удобным.
- Большая библиотека компонентов и возможность добавления новых.
- Поддержка многослойных плат и гибкая система маршрутизации.
- Высокая точность моделирования и анализа с помощью встроенных инструментов симуляции и анализа.
- Высокий уровень автоматизации процессов проектирования, что позволяет значительно сократить время и затраты на проектирование.
- Поддержка различных операционных систем.

Минусы KiCAD:

- Низкая скорость работы в сравнении с коммерческими системами.
- Ограниченные возможности по созданию 3D-моделей и обеспечению взаимодействия между компонентами.
- Нет встроенных инструментов для создания макета печатной платы (PCB).
- Ограниченная интеграция с некоторыми другими инструментами и процессами проектирования.

Proteus – это коммерческая САПР для разработки электронных схем и печатных плат, разработанная компанией Labcenter Electronics. Она предоставляет широкий спектр инструментов для проектирования и моделирования электронных схем и печатных плат, а также встроенные инструменты симуляции.

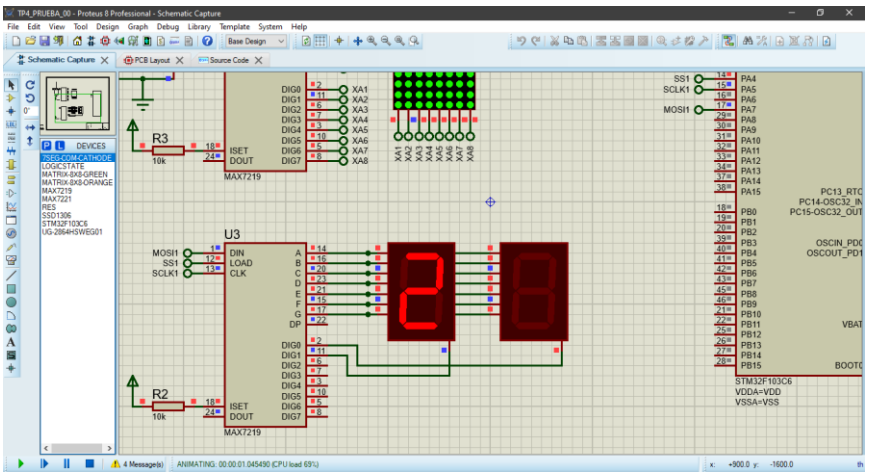


Рисунок 2.2.3 – Интерфейс САПР Proteus

Плюсы Proteus:

- Интуитивно понятный пользовательский интерфейс и удобное управление проектами.
- Широкий спектр инструментов для проектирования и моделирования электронных схем и печатных плат.
- Возможность импортировать и экспортировать данные в различных форматах, включая SPICE и STEP.
- Поддержка многопроцессорных систем, что обеспечивает более быструю работу с большими проектами.
- Встроенные инструменты симуляции, такие как PSpice и ISIS, позволяют проводить анализ и оптимизацию проектов.
- Возможность создания 3D-моделей печатных плат и компонентов.
- Большая библиотека компонентов и возможность добавления новых.

Минусы Proteus:

- Высокая стоимость лицензии, особенно для коммерческих проектов.
- Не всегда достаточно гибкие настройки автоматической маршрутизации печатных плат.
- Ограниченная поддержка для некоторых компонентов и форматов файлов.
- Ограниченная интеграция с некоторыми другими инструментами и процессами проектирования.

В рамках курсового проекта по разработке принципиальной электрической схемы KiCAD представляется хорошим выбором САПР. Она предоставляет бесплатный и открытый исходный код, что делает ее доступной для всех. В то же время, KiCAD обладает всеми необходимыми функциями для разработки принципиальных схем и печатных плат, включая библиотеку компонентов, инструменты для автоматической и ручной маршрутизации, а также встроенный инструмент для симуляции.

2.2.2 Описание используемых модулей и библиотечных элементов

Ранее была описана структурная схема мобильного проигрывателя аудио файлов, где все устройство разбиты на основные компоненты для его работы:

- Отладочная плата Arduino Nano V3;
- Модуль для чтения/записи данных с SD-карты HW-125;
- Аудио усилитель на чипе LM386;
- Модуль зарядки на чипе TP4056;
- Модуль DC-DC повышающий на чипе MT3608.

Каждый отдельный компонент проигрывателя представляет собой отдельный модель или плату и имеет свою принципиальную электрическую схему.

Модуль зарядки на чипе TP4056. Модуль разработан на базе чипа TP4056.

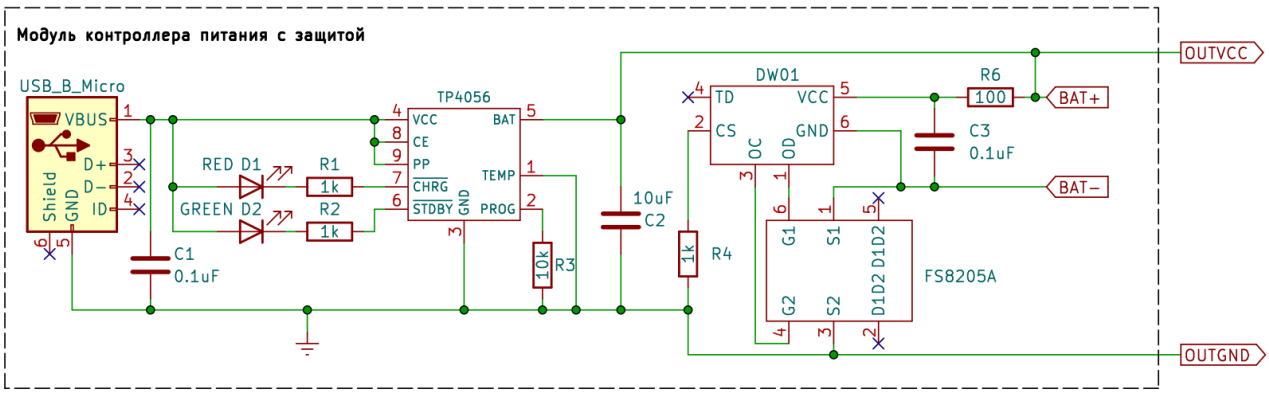


Рисунок 2.2.4 – Принципиальная схема модуля зарядки на чипе TP4056

TP4056 – это комплексное линейное зарядное устройство постоянного тока или постоянного напряжения для литий-ионных аккумуляторов одной ячейки.

Помимо основного чипа, имеет чип DW01, предназначенный для защиты аккумулятора от повреждения или сокращения срока службы из-за перезаряда, переразряда и/или перегрузки по току. Также имеет двойной N-канальный MOSFET транзистор FS8205A.

Так как выходное напряжение будет равно примерно до 3.7V, необходимо использовать DC-DC повышающий преобразователь напряжения, как уже было описано. В рамках курсового проекта будет использоваться **DC-DC повышающий преобразователь напряжения HW-045**, построенный на однокристальном чипе регулятора MT3608. Эта микросхема содержит в себе несколько функциональных блоков, включая силовой MOSFET-транзистор, контроллер управления ШИМ-модуляцией, частотный генератор и управляющую логику.

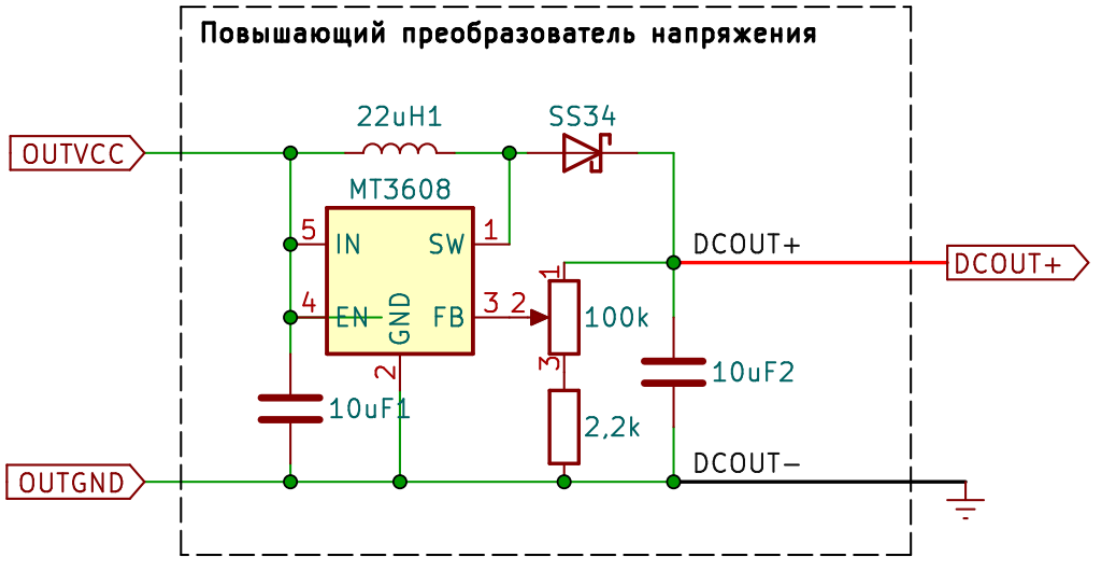


Рисунок 2.2.5 – Принципиальная схема повышающий преобразователя HW-045

MT3608 — это повышающий преобразователь с 6-выводным током SOT23 с постоянной частотой, предназначенный для небольших приложений с низким энергопотреблением.

Для усиления звукового сигнала будет использоваться **модуль аудио усилителя на базе LM386**. Модуль аудио усилителя LM386 – это одноканальный аудио усилитель общего назначения, который может работать с мощностью до 325 мВт.

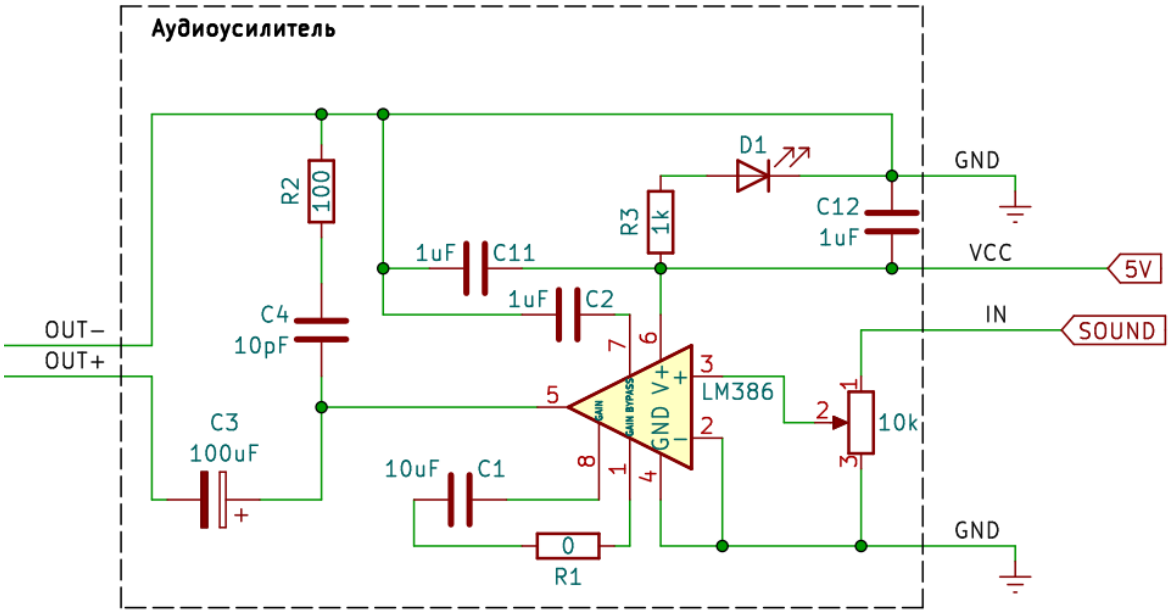


Рисунок 2.2.6 – Принципиальная схема аудио усилителя на базе LM386

LM386 – это усилитель мощности, предназначенный для использования в низковольтных потребительских устройствах.

Для чтения данных с SD-карты необходим специальный модуль. На общей принципиальной схеме изображен **компонент SD_Card_Module** из библиотеки Charleslabs_Parts. Данный компонент представляет собой аналог реального модуля используемого в проекте.

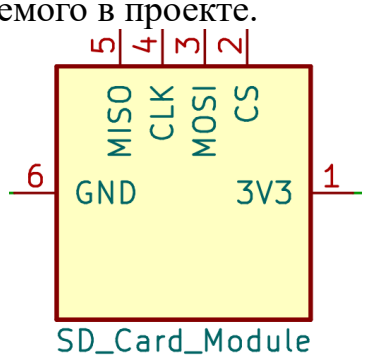


Рисунок 2.2.7 – Компонент SD_Card_Module

Управляться проигрыватель будет микропроцессором ATMega328P, установленном на отладочной плате Arduino Nano V3, на общей принципиальной схеме представленной в виде компонента Arduino_Nano_v3.x из библиотеки MCU_Module.

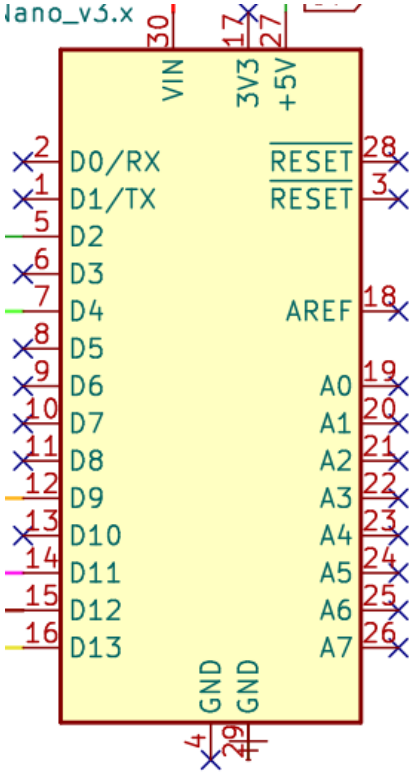


Рисунок 2.2.8 – Компонент Arduino_Nano_v3.x

3 Разработка алгоритма функционирования мобильного проигрывателя аудио файлов

3.1 Описание алгоритма функционирования мобильного проигрывателя аудио файлов

Начать стоит с технических требований к алгоритму. Стоит определить функции, которые должны выполняться.

В рамках курсового проекта у мобильного проигрывателя будет одна кнопка, следовательно все управление должно осуществляться одной кнопкой. Аудио файлы должны считываться с SD-карты, при этом файлы находятся в папке «music».

Исходя из вводных данных можно следующий необходимый для работы функционал:

- При коротком нажатии на кнопку воспроизведение аудио файла должно ставиться на паузу, при повторном коротком нажатии – должно продолжиться;
- При долгом нажатии на кнопку должно начинаться проигрывание следующего аудиофайла;
- Если завершилось проигрывание какого-либо аудио файла автоматически должно начаться проигрывание следующего;

3.2 Выбор IDE для разработки

IDE (Integrated Development Environment) – это программное обеспечение, которое позволяет разработчикам создавать, отлаживать и тестировать программы в одной интегрированной среде. IDE обычно включает в себя текстовый редактор для написания кода, средства отладки для исправления ошибок, компиляторы и интерпретаторы для преобразования исходного кода в исполняемый файл, а также другие инструменты для повышения производительности и эффективности разработки программного обеспечения. IDE часто используются в различных языках программирования, таких как Java, Python, C ++ и других.

Существует несколько IDE для программирования платформы Arduino, которые пользуются популярностью среди разработчиков:

- Arduino IDE;
- Visual Studio Code;
- PlatformIO;
- Eclipse IDE;
- CodeBlocks.

3.2.1 Описание Arduino IDE

Arduino IDE – это официальная интегрированная среда разработки, которая разрабатывается командой Arduino. Она предназначена для программирования платформы Arduino и доступна для Windows, Mac и Linux.

Основные особенности Arduino IDE:

- Простота использования: Arduino IDE имеет простой интерфейс и низкий порог входа для новичков в программировании.
- Совместимость с Arduino-платами: Arduino IDE может работать с большинством платформ и моделей Arduino.
- Встроенный текстовый редактор: Arduino IDE предоставляет инструменты для создания и редактирования кода.
- Библиотеки: Arduino IDE поставляется с широким спектром библиотек, которые содержат множество готовых функций, упрощающих разработку проектов.
- Загрузчик: Arduino IDE имеет встроенный загрузчик, который позволяет загружать код на Arduino-платы без необходимости использовать отдельное ПО.
- Отладка: Arduino IDE имеет встроенный отладчик, который может помочь в идентификации и устранении ошибок.

Плюсы Arduino IDE для программирования Arduino:

- Легкость в использовании: Arduino IDE имеет простой интерфейс и низкий порог входа для новичков в программировании.
- Совместимость: Arduino IDE поддерживает большинство платформ и моделей Arduino.
- Библиотеки: Arduino IDE поставляется с широким спектром библиотек, которые содержат множество готовых функций, упрощающих разработку проектов.
- Загрузчик: Arduino IDE имеет встроенный загрузчик, который упрощает загрузку кода на Arduino-платы.

Минусы Arduino IDE для программирования Arduino:

- Ограниченность по функционалу: Arduino IDE имеет ограниченные возможности по сравнению с более продвинутыми IDE
- Отсутствие функций автодополнения и подсветки синтаксиса: Arduino IDE не обладает некоторыми функциями, которые могут быть полезными для разработчиков.
- Ограничения по расширяемости: Arduino IDE имеет ограниченные возможности для добавления дополнительных библиотек и инструментов.

- В целом, Arduino IDE является простой в использовании и надежной IDE для разработки Arduino-проектов. Она имеет все необходимые инструменты для создания и загрузки кода на Arduino-платы.

3.2.2 Описание Visual Studio Code

Visual Studio Code (VS Code) – это бесплатная кросс-платформенная интегрированная среда разработки (IDE) от Microsoft. VS Code предоставляет возможность разрабатывать приложения на многих языках программирования, включая C++, Python, JavaScript, и, конечно же, Arduino. VS Code обладает множеством функциональных возможностей, которые делают ее одной из самых популярных IDE для программирования вообще и для программирования Arduino в частности.

Преимущества VS Code для программирования Arduino:

- Бесплатность: VS Code является бесплатной IDE и не требует покупки лицензии, что делает ее доступной для всех разработчиков.
- Кроссплатформенность: VS Code работает на Windows, Mac и Linux, что позволяет использовать ее на любой платформе.
- Расширяемость: Возможности VS Code можно расширять за счет использования различных плагинов и расширений, доступных в магазине расширений. Существуют расширения, специально разработанные для программирования Arduino, которые облегчают работу разработчика.
- Отладка: VS Code предоставляет мощный отладчик, который позволяет проверять работу кода на Arduino в режиме реального времени, а также удобную отладочную консоль.
- Интеграция с Git: VS Code предоставляет удобный интерфейс для работы с системой контроля версий Git, что делает процесс работы с кодом более простым и удобным.
- Наличие множества инструментов: VS Code предоставляет широкий выбор инструментов для редактирования и форматирования кода, а также удобную навигацию по файлам и проектам.

Недостатки VS Code для программирования Arduino:

- Сложность настройки: Некоторые настройки могут потребовать дополнительного времени на изучение документации, но это компенсируется более продвинутым функционалом.
- Потребление ресурсов: VS Code потребляет больше ресурсов компьютера, чем некоторые другие IDE, что может замедлить работу на медленных компьютерах.

В целом, VS Code - это мощная и удобная IDE для программирования Arduino, которая позволяет ускорить процесс разработки и улучшить качество кода. Она обладает достаточным функционалом для создания качественного и профессионального кода, а также предоставляет возможности для интеграции

3.2.3 Описание PlatformIO

PlatformIO – это кроссплатформенная интегрированная среда разработки (IDE) для создания встраиваемых систем, включая Arduino. Она предоставляет широкий набор инструментов для разработки, отладки и загрузки кода на микроконтроллеры. PlatformIO является более продвинутой и мощной альтернативой официальной Arduino IDE.

Некоторые из основных возможностей и преимуществ PlatformIO для программирования Arduino включают

- Поддержка большого количества платформ и микроконтроллеров - PlatformIO поддерживает более 600 платформ и более 60 производителей микроконтроллеров, включая Arduino, ESP32, ESP8266, STM32 и многие другие.
- Расширенная библиотека - PlatformIO предоставляет доступ к более чем 30 000 библиотекам для Arduino, которые можно устанавливать и использовать в своих проектах.
- Широкий выбор инструментов - PlatformIO включает в себя множество инструментов для разработки и отладки кода, включая интегрированную консоль, терминал, автодополнение, плагин для отладки и другие.
- Удобный интерфейс - PlatformIO обладает более современным и удобным интерфейсом, чем официальная Arduino IDE, который упрощает работу с кодом и инструментами.
- Кроссплатформенность - PlatformIO поддерживает работу на Windows, macOS и Linux, что облегчает работу разработчиков, которые используют разные операционные системы.

Некоторые из недостатков PlatformIO для программирования Arduino включают:

- Сложность настройки - PlatformIO имеет более сложный процесс установки и настройки, чем официальная Arduino IDE.
- Большой объем установки - PlatformIO требует больше места на диске, чем официальная Arduino IDE, из-за большого количества инструментов и библиотек.
- Более высокие требования к системе - из-за большого количества инструментов и библиотек, PlatformIO может требовать более мощной системы, чем официальная Arduino IDE.

В целом, PlatformIO является мощной альтернативой для программирования Arduino, которая предоставляет широкий выбор инструментов и библиотек, а также поддерживает большое

3.3 Описание используемых библиотек

При написании кода были использованы следующие библиотеки:
- SPI.h;
- SD.h;
- TMRpcm.h.

3.3.1 Описание библиотеки SPI.h

Библиотека SPI.h предназначена для работы с последовательным интерфейсом обмена данными SPI (Serial Peripheral Interface) на микроконтроллерах Arduino и других подобных устройствах. SPI является популярным интерфейсом для связи между микроконтроллерами и другими устройствами, такими как сенсоры, экраны, радиомодули и другие периферийные устройства. [10]

SPI использует синхронную передачу данных, где один устройство является мастером, который инициирует обмен, а другие устройства являются слейвами, которые отвечают на запросы мастера. Библиотека SPI.h позволяет микроконтроллеру Arduino работать как мастер в интерфейсе SPI, а также обмениваться данными с другими устройствами. [10]

В библиотеке SPI.h доступны функции для инициализации интерфейса SPI, установки режима работы (CPOL и CPHA), настройки скорости передачи данных, отправки и приема данных по интерфейсу SPI.

Некоторые из функций, доступных в библиотеке SPI.h, включают:
- *SPI.begin()*: инициализирует интерфейс SPI на микроконтроллере и устанавливает режим работы (мастер или слейв).
- *SPI.setClockDivider()*: устанавливает скорость передачи данных по интерфейсу SPI.
- *SPI.setDataMode()*: устанавливает режим работы интерфейса SPI (CPOL и CPHA).
- *SPI.transfer()*: отправляет и/или принимает данные по интерфейсу SPI.

Библиотека SPI.h является частью стандартной библиотеки Arduino и поставляется вместе с Arduino IDE. Она легко доступна и проста в использовании, что делает ее популярным выбором для работы с интерфейсом SPI на платформе Arduino. В рамках курсового проекта будет использоваться для работы библиотеки SD.h.

3.3.2 Описание библиотеки SD.h

Библиотека SD.h – это стандартная библиотека Arduino для работы с SD-картами. Она позволяет осуществлять чтение и запись файлов на SD-карту, используя стандартный SPI интерфейс. [11]

Некоторые из основных функций библиотеки SD.h включают:

- *SD.begin()* – инициализирует библиотеку и устанавливает соединение с SD-картой.
- *SD.open()* – открывает файл на SD-карте для чтения или записи.
- *file.read()* – читает данные из файла на SD-карте.
- *file.write()* – записывает данные в файл на SD-карте.
- *file.close()* – закрывает файл на SD-карте.

Библиотека SD.h также имеет некоторые дополнительные функции, такие как проверка наличия SD-карты, проверка свободного места на карте и т.д.

Несмотря на то, что библиотека SD.h является стандартной библиотекой для работы с SD-картами, ее использование требует определенных навыков и знаний. Например, при работе с SD-картами необходимо учитывать ограничения скорости чтения и записи, а также заботиться о надежности хранения данных на карте.

В целом, библиотека SD.h предоставляет удобный и простой способ работы с SD-картами на платформе Arduino. В рамках курсового проекта данная библиотека будет использоваться для аудио файлов с SD-карты и для работы библиотеки TMRpcm.h.

3.3.3 Описание библиотеки TMRpcm.h

Библиотека TMRpcm.h представляет собой библиотеку, предоставляющую простой способ воспроизведения звуковых файлов на платформе Arduino без использования дополнительного аппаратного обеспечения, такого как MP3-декодер или звуковая карта. Она позволяет воспроизводить звуковые файлы в формате WAV с частотой дискретизации 8, 16 и 32 кГц. [12]

Для использования библиотеки TMRpcm.h, необходимо подключить библиотеку и установить соответствующие настройки для воспроизведения звуковых файлов. Она может быть использована для воспроизведения звуковых эффектов, голосовых уведомлений или музыки в проектах Arduino.

Некоторые из основных функций библиотеки TMRpcm.h включают:

- play - воспроизведение звукового файла с заданным именем файла.
- setVolume - установка громкости воспроизведения.
- isPlaying - проверка, воспроизводится ли в данный момент звуковой файл.

Недостатком этой библиотеки является то, что она использует значительное количество памяти Arduino для буферизации звуковых файлов, что может привести к ограничению доступной памяти для других частей программы. Также она не поддерживает воспроизведение звуковых файлов с более высокой частотой дискретизации, что может ограничить выбор звуковых файлов для воспроизведения.

В целом, библиотека TMRpcm.h является удобным и простым способом воспроизведения звуковых файлов на платформе Arduino, который подходит для простых проектов, которые не требуют высокой частоты дискретизации звука и большого объема буфера.

3.4 Описание кода и функций алгоритма

В самом начале подключаются необходимые библиотеки:

```
#include <SPI.h> //SPI библиотека для SD карты
#include <SD.h> //библиотека чтобы считывать информацию с SD карты
#include "TMRpcm.h" //библиотека чтобы проигрывать аудио файлы
```

Рисунок 3.4.1 – Фрагмент кода (подключение библиотек)

После чего объявляется константы:

```
#define SD_CHIP_SELECT 4 // вывод, подключенный к линии выбора чипа на SD карте
#define BUTTON_PIN 2 // пин, к которому подключена кнопка
#define AUDIO_PIN 9 // пин, на который вызодит аудио сигнал
```

Рисунок 3.4.2 – Фрагмент кода (объявление констант)

После объявление констант, необходимо объявить необходимые для работы переменные:

```
TMRpcm music; // объявление объекта библиотеки для проигрываения аудио файлов

boolean button  = false; // Логическая переменная для хранения состояние кнопки
boolean press_flag = false; // Флажок нажатия кнопки
boolean long_press_flag = false; // Флажок долгого нажатия на кнопку
unsigned long last_press = 0; // Хранит время с последнего нажатия на кнопку

String song_name_str = ""; // переменная для хранения имени аудио файла
char buf[13]; // переменная для приведения к типу char[]
boolean play_pause = true; // переменная для хранения состояние play/pause

File root; // объявление объекта файла для чтения данных с SD-карты
```

Рисунок 3.4.3 – Фрагмент кода (объявление переменных)

В стандартной функции *setup()* задаются начальные значения некоторых переменных, задаются некоторые параметры для дальнейшей работы и происходит инициализация SD-карты:

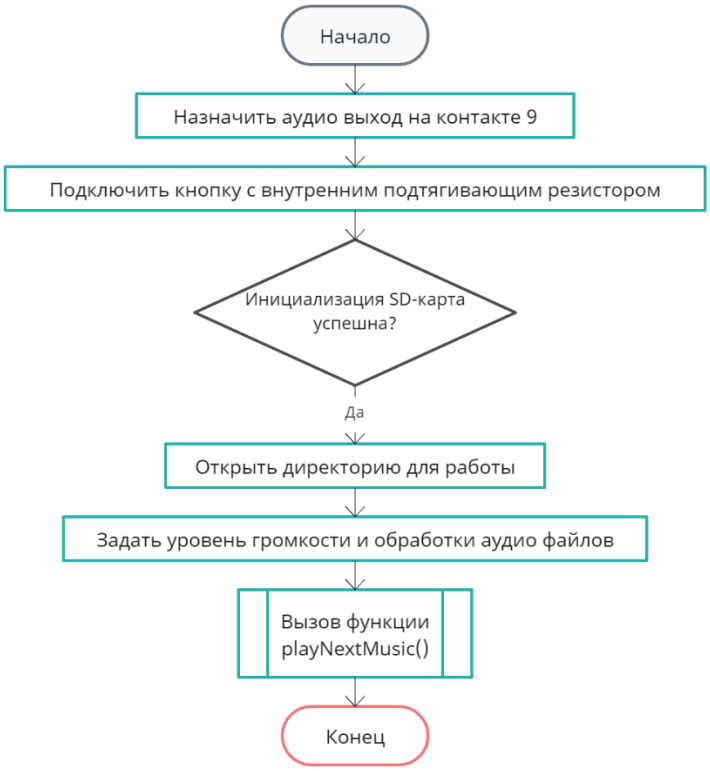


Рисунок 3.4.4 – Блок-схема алгоритма функции setup

```
void setup() {

  music.speakerPin = AUDIO_PIN; // аудио выход на контакте 9

  pinMode(BUTTON_PIN, INPUT_PULLUP); // подключение кнопки с внутренним
      подтягивающим резистором

  if (!SD.begin(SD_ChipSelect)) { // инициализация и проверка SD-карты
    while (true);
  }
  root = SD.open("/music"); // открыть директорию для работы

  music.setVolume(4); // задать необходимый уровень громкости
  music.quality(2); // задать необходимый уровень обработки аудио файлов

  playNextMusic(); // вызываем функуции для включения проигрываения аудио файла,
      при включении проигрывателя

}
```

Рисунок 3.4.5 – Фрагмент кода (функция setup)

Функция, в которой происходит чтение следующего файла и проверка его наличия *playNextMusic()*:

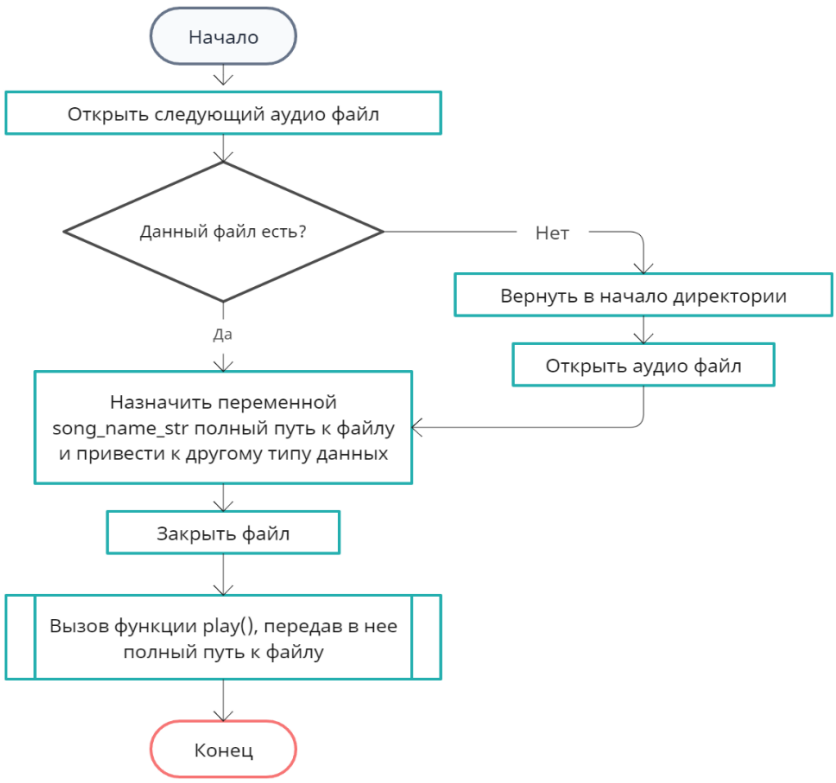


Рисунок 3.4.6 – Блок-схема алгоритма функции playNextMusic

```
void playNextMusic() {

  File next_song = root.openNextFile(); // открываем следующий файл в директории

  if (!next_song) { // проверяем наличие этого файла
    root.rewindDirectory(); // если файла нет, возвращаемся в начало директории
    next_song = root.openNextFile(); // открываем файл в директории
  }

  song_name_str = "/music/" + next_song.name();
  song_name_str.toCharArray(buf, song_name_str.length() + 1);

  next_song.close();

  music.play(buf); // воспроизводим файл

}
```

Рисунок 3.4.7 – Фрагмент кода (функция playNextMusic)

Исходя из требований к алгоритму необходимо при коротком нажатии на кнопку поставить воспроизведение на паузу, что реализовано в функции *isButtonSingle()*, также при длительном нажатии на кнопку начать проигрывание следующего файла, что реализовано в функции *isButtonHold()*:

```
// функция вызывающаяся при коротом нажатии на кнопку
void isButtonSingle() {
  play_pause = !play_pause;
  music.pause();
}

// функция вызывающаяся при продолжительном нажатии на кнопку
void isButtonHold() {
  playNextMusic();
}
```

Рисунок 3.4.8 – Фрагмент кода (функции isButtonSingle и isButtonHold)

В заключение главная функция, где происходит проверка нажатия кнопки, длительности нажатия кнопки, закончилось ли воспроизведения файла. Функция *loop()* выполняется бесконечно, пока работает устройство.

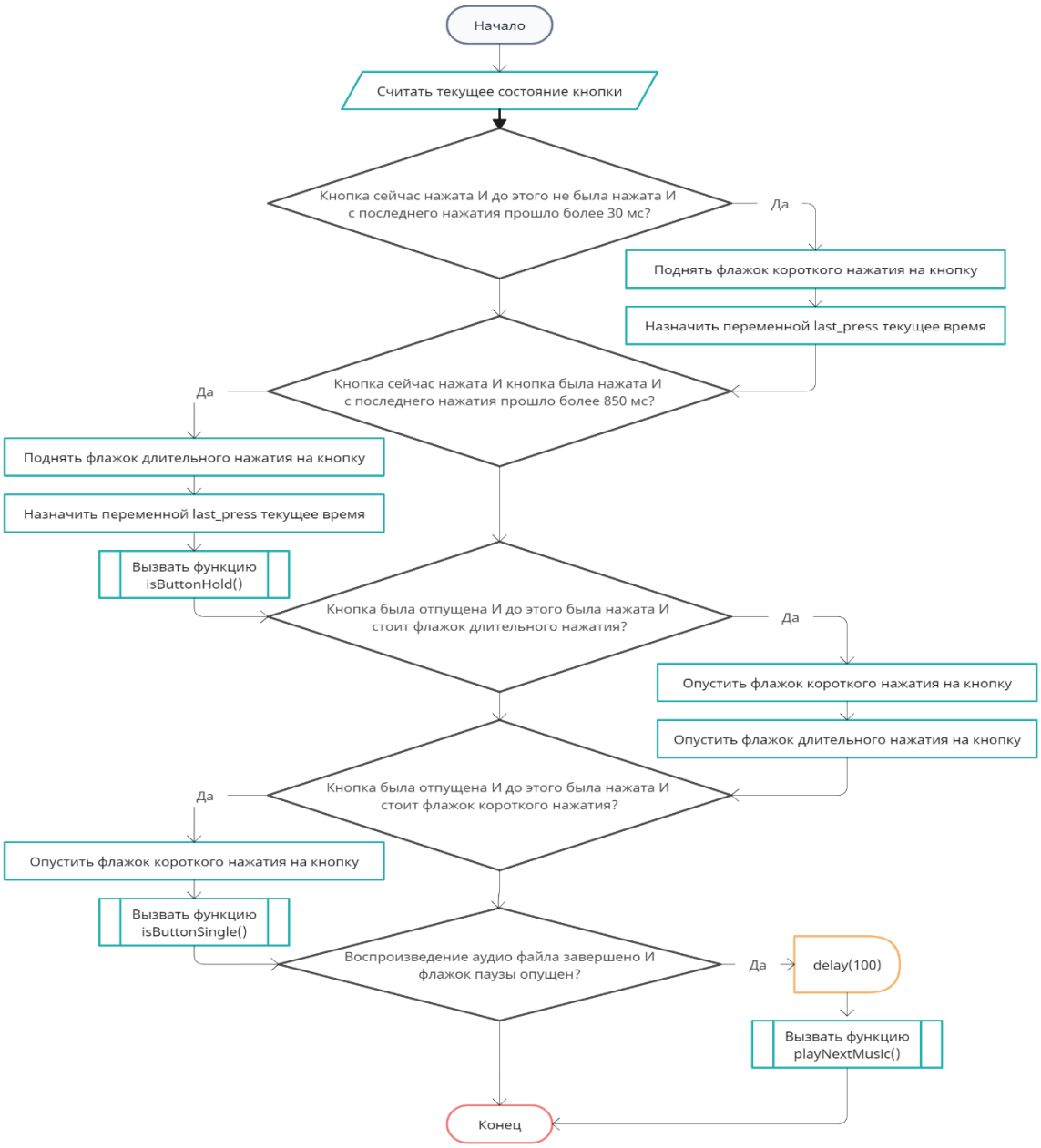


Рисунок 3.4.9 – Блок-схема алгоритма функции loop

```
void loop() {
  button = !digitalRead(BUTTON_PIN);    // считывает текущее состояние кнопки

  // обработка короткого нажатия и устранение дребезга
  if (button == true  && press_flag == false && millis() - last_press > 30) {
    press_flag = !press_flag;
    last_press = millis();
  }
  //обработка длительного нажатия
  if (button == true && press_flag == true && millis() - last_press > 850) {
    long_press_flag = !long_press_flag;
    last_press = millis();
    isButtonHold();
  }
  if (button == false && press_flag == true && long_press_flag == true) {
    press_flag = !press_flag;
    long_press_flag = !long_press_flag;
  }
  if (button == false && press_flag == true && long_press_flag == false) {
    press_flag = !press_flag;
    isButtonSingle();
  }

  // если заканчивается воспроизвдение файла, воспроизвести следующего файла
  if (!music.isPlaying() && play_pause) {
    delay(100);
    playNextMusic();
  }
}
```

Рисунок 3.4.7 – Фрагмент кода (функции loop)

Заключение

По ходу выполнения курсового проекта по разработке мобильного проигрывателя аудио файлов был проведен детальная анализ предметной области и принципы работы проигрывателя аудио файлов. В результате было спроектировано и создано устройство, которое позволяет пользователям воспроизводить аудио файлы с SD-карты с выходом на miniJack.

В процессе анализа предметной области были рассмотрены различные форматы аудио файлов и их особенности. Были изучены различные алгоритмы обработки звука, был изучен принцип обработки звука и характеристики цифрового звука. Были изучены принципы работы аппаратной платформы Arduino, схемы автономного питания от аккумулятора с возможностью его подзарядки и работы аудио усилителя.

В итоге были разработаны принципиальная электрическая схема, структурная электрическая схема и алгоритм работы мобильного проигрывателя аудио файлов. Кроме того, устройство было собрано и опробовано.

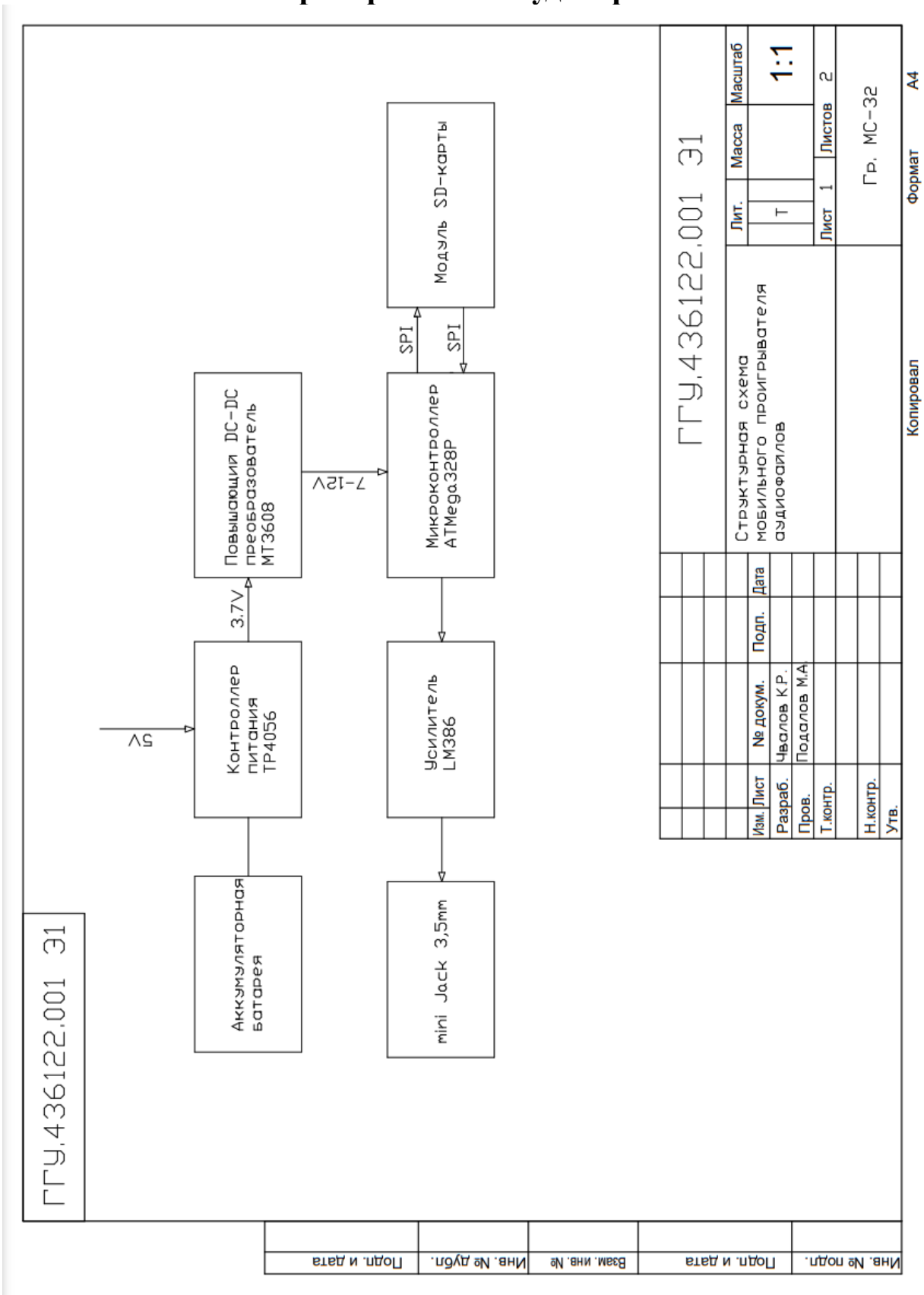
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Цифровые аудиоформаты [Электронный ресурс] // Свободная энциклопедия Википедия. – URL: https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D1%8B%D0%B5_%D0%B0%D1%83%D0%B4%D0%B8%D0%BE%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D1%8B. – Дата доступа: 27.03.2023.
2. Цифровое представление аналогового аудиосигнала. Краткий ликбез [Электронный ресурс] // Хабр. – URL: https://habr.com/ru/articles/503786/. – Дата доступа: 27.03.2023.
3. Цифровая звукозапись [Электронный ресурс] // Свободная энциклопедия Википедия. – URL: https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B7%D0%B2%D1%83%D0%BA%D0%BE%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C. – Дата доступа: 27.03.2023.
4. Преимущества и недостатки популярных аудиоформатов [Электронный ресурс] // PULT.RU. – URL: https://www.pult.ru/articles/interesting/audioformaty/. – Дата доступа: 29.03.2023.
5. Arduino [Электронный ресурс] // Свободная энциклопедия Википедия. – URL: https://ru.wikipedia.org/wiki/Arduino. – Дата доступа: 31.03.2023.
6. ESP8266 [Электронный ресурс] // Свободная энциклопедия Википедия. – URL: https://ru.wikipedia.org/wiki/ESP8266. – Дата доступа: 31.03.2023.
7. Raspberry Pi [Электронный ресурс] // Свободная энциклопедия Википедия. – URL: https://ru.wikipedia.org/wiki/Raspberry_Pi. – Дата доступа: 27.03.2023.
8. Классы усилителей [Электронный ресурс] // POP-MUSIC. – URL: https://pop-music.ru/articles/klassy-usiliteley/. – Дата доступа: 01.04.2023.
9. Модуль аудио усилитель звука LM386 монофонический 2Вт [Электронный ресурс] // Compact Tool. – URL: https://compacttool.ru/modul-audio-usilitel-zvuka-lm386-monofonicheskiy-2vt. – Дата доступа: 15.04.2023.
10. Библиотека SPI [Электронный ресурс] // arduino.ua. – URL: https://doc.arduino.ua/ru/prog/SPI. – Дата доступа: 30.04.2023.
11. Библиотека SD для работы Arduino с SD картами [Электронный ресурс] // Radio Prog. – URL: https://radioprog.ru/post/334. – Дата доступа: 30.04.2023.

12. TMRpcm: Arduino библиотека для воспроизведения PCM/WAV аудиофайлов напрямую с SD карты [Электронный ресурс] // Radio Prog. – URL: https://radioprog.ru/post/270. – Дата доступа: 30.04.2023.

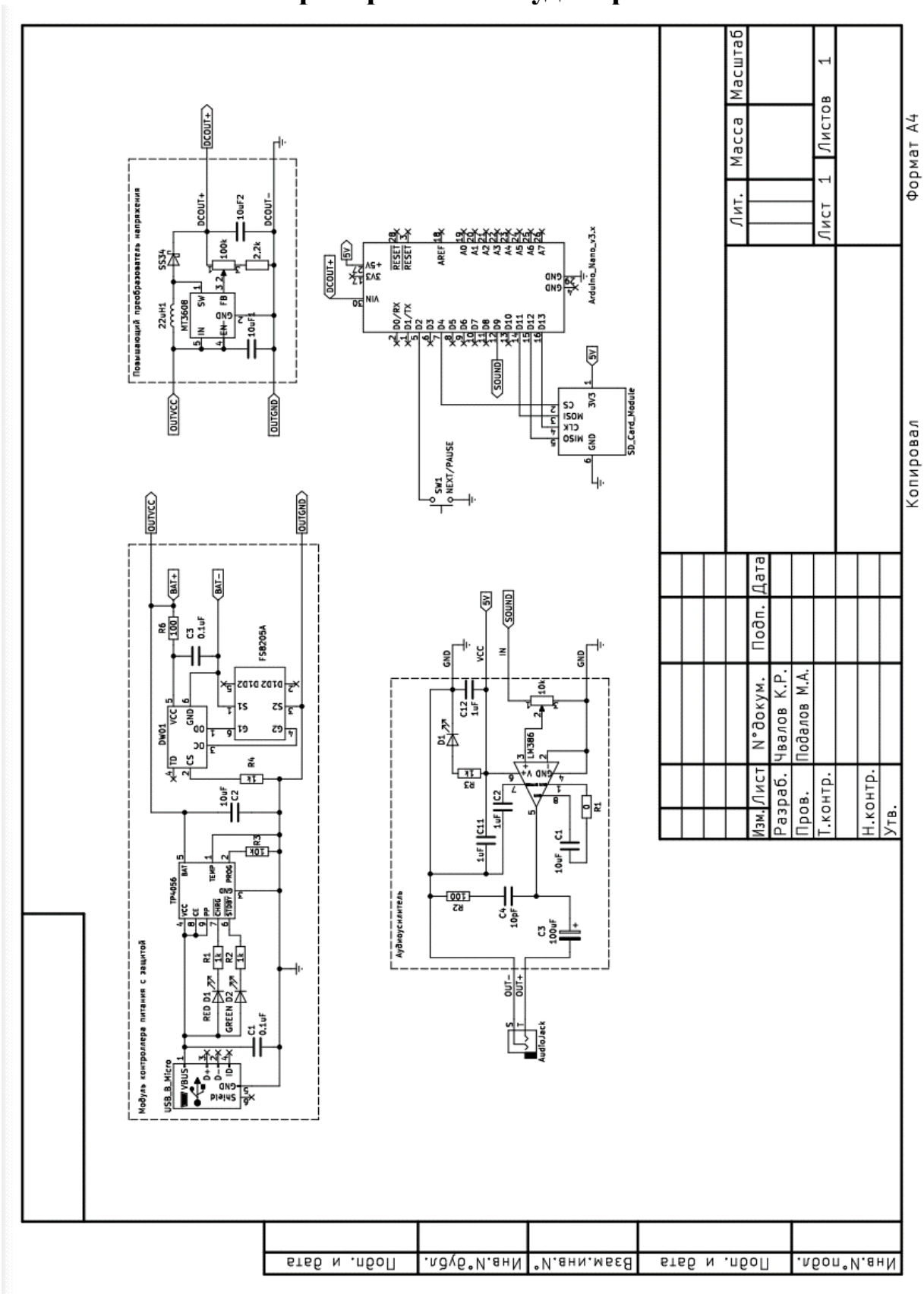
ПРИЛОЖЕНИЕ А

Структурная электрическая схема мобильного проигрывателя аудио файлов



ПРИЛОЖЕНИЕ Б

Принципиальная электрическая схема мобильного проигрывателя аудио файлов



ПРИЛОЖЕНИЕ В
Код алгоритма мобильного проигрывателя аудио файлов

```
#include <SPI.h> //SPI библиотека для SD карты
#include <SD.h> //библиотека чтобы считывать информацию с SD карты
#include "TMRpcm.h" //библиотека чтобы проигрывать аудио файлы

#define SD_CHIP_SELECT 4 // вывод, подключенный к линии выбора чипа на SD карте
#define BUTTON_PIN 2 // пин, к которому подключена кнопка
#define AUDIO_PIN 9 // пин, на который вызодит аудио сигнал

TMRpcm music; // объявление объекта библиотеки для проигрываения аудио файлов

boolean button  = false; // Логическая переменная для хранения состояние кнопки
boolean press_flag = false; // Флажок нажатия кнопки
boolean long_press_flag = false; // Флажок долгого нажатия на кнопку
unsigned long last_press = 0; // Хранит время с последнего нажатия на кнопку

String song_name_str = ""; // переменная для хранения имени аудио файла
char buf[13]; // переменная для приведения к типу char[]
boolean play_pause = true; // переменная для хранения состояние play/pause

File root; // объявление объекта файла для чтения данных с SD-карты

void setup() {
  music.speakerPin = AUDIO_PIN; // аудио выход на контакте 9

  pinMode(BUTTON_PIN, INPUT_PULLUP); // подключение кнопки с внутренним
подтягивающим резистором

  if (!SD.begin(SD_ChipSelect)) { // инициализация и проверка SD-карты
    while (true);
  }
  root = SD.open("/music"); // открыть директорию для работы

  music.setVolume(4); // задать необходимый уровень громкости
  music.quality(2); // задать необходимый уровень обработки аудио файлов

  playNextMusic(); // вызываем функуции для включения проигрываения аудио файла,
при включении проигрывателя
}

void loop() {
  button = !digitalRead(BUTTON_PIN);   // считывает текущее состояние кнопки

  // обработка короткого нажатия и устранение дребезга
  if (button == true  && press_flag == false && millis() - last_press > 30) {
    press_flag = !press_flag;
    last_press = millis();
```

```
  }
  //обработка длительного нажатия
  if (button == true && press_flag == true && millis() - last_press > 850) {
    long_press_flag = !long_press_flag;
    last_press = millis();
    isButtonHold();
  }
  if (button == false && press_flag == true && long_press_flag == true) {
    press_flag = !press_flag;
    long_press_flag = !long_press_flag;
  }
  if (button == false && press_flag == true && long_press_flag == false) {
    press_flag = !press_flag;
    isButtonSingle();
  }

  // если заканчивается воспроизвдение файла, воспроизвести следующего файла
  if (!music.isPlaying() && play_pause) {
    delay(100);
    playNextMusic();
  }
}

// функция вызывающаяся при коротом нажатии на кнопку
void isButtonSingle() {
  play_pause = !play_pause;
  music.pause();
}

// функция вызывающаяся при продолжительном нажатии на кнопку
void isButtonHold() {
  playNextMusic();
}

void playNextMusic() {
  File next_song = root.openNextFile(); // открываем следующий файл в директории

  if (!next_song) { // проверяем наличие этого файла
    root.rewindDirectory(); // если файла нет, возвращаемся в начало директории
    next_song = root.openNextFile(); // открываем файл в директории
  }

  song_name_str = "/music/" + next_song.name();
  song_name_str.toCharArray(buf, song_name_str.length() + 1);

  next_song.close();

  music.play(buf); // воспроизводим файл
}
```