Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Схемотехника

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

АНАЛИЗАТОР ЗВУКОВОГО СИГНАЛА

БГУИР КП 1-40 02 01 104 ПЗ

Студент: группы 150504,
Горбачевский К. В.

Руководитель:
Калютчик А. А.

Минск 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____
(подпись)
«_____»_____2021 г.

З А Д А Н И Е
по курсовому проектированию

Студенту Горбачевскому Кириллу Витальевичу
(фамилия, имя, отчество)

1. Тема проекта Анализатор звукового сигнала

2. Срок сдачи студентом законченного проекта с 06.12.2021 по 09.12.2021

3. Исходные данные к проекту:

1. Микроконтроллер – тактовая частота не менее 20 кГц, не менее 10 входов/выходов.
2. Источник питания – напряжение 9 В, максимальный выходной ток не менее 3 А.
3. Модуль усиления – напряжение питания 7 – 36 В, коэфициент усиления не меньше 10.
4. Дисплей – напряжение питания 3.3-5 В, разрешение не менее 128x64.

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Введение
1.Обзор литературы.
2.Разработка структуры микропроцессорного радиоуправляемого передвижного устройства.
3.Обоснование выбора узлов, элементов функциональной схемы устройства.
4. Разработка принципиальной электрической схемы устройства.
5. Разработка программного обеспечения.
Заключение.

Литература.

5. Перечень графического материала (с точным указанием обязательных чертежей)

1. Структурная схема анлизатора звукового сигнала устройства (формат А4)
2. Функциональная электрическая схема анлизатора звукового сигнала (формат А3)
3. Принципиальная электрическая схема анлизатора звукового сигнала (формат А3)

6. Консультант по проекту (с назначением разделов проекта) А.А. Калютчик

7. Дата выдачи задания 10.09.2023

8. Календарный график работы над проектом на весь период проектирования (с назначением сроков исполнения и трудоемкости отдельных этапов):

разделы 1,2 к 24.09 – 20 %;
раздел 3 к 15.10 – 20 %;
раздел 4 к 05.11 – 25 %;
раздел 5 к 19.11 – 20 %;
оформление пояснительной записки и графического материала к 06.12 – 15 %;
защита курсового проекта с 07.12 по 14.12.

РУКОВОДИТЕЛЬ_____асистент каф. ЭВМ Калютчик А.А.
(подпись)

Задание принял к исполнению 10.09.2023_____К. В. Горбачевский
(дата и подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Темой данного курсового проекта является разработка устройства, анализирующего аналоговый сигнал на базе микроконтроллера.

Современная технологическая эпоха обогатила мир музыки и музыкантов инновационными средствами и инструментами для творчества. Однако, несмотря на множество цифровых достижений в области музыкального оборудования, аналоговые инструменты, такие как гитары, остаются популярными среди музыкантов разного уровня мастерства.

Важным аспектом игры на аналоговых инструментах, в том числе на гитаре, является настройка для достижения чистых и гармоничных звуков. В данном контексте представляется актуальной тема создания устройств, позволяющих музыкантам легко и точно настраивать свои инструменты.

Целью данного курсового проекта является разработка и реализация анализатора аналогового сигнала на базе микроконтроллера, предназначенного для настройки гитары.

Данное устройство может быть использовано не только для анализа сигнала исходящего от электрогитары. Прибор сможет определять частоту (тон) любого сигнала, передаваемого по кабелю Jack 6.3

В данной работе будет рассмотрен весь процесс разработки и реализации анализатора аналогового сигнала, начиная с выбора аппаратной платформы и заканчивая созданием программного обеспечения для обработки сигнала. Также будут рассмотрены различные методы анализа и алгоритмы, применяемые для определения настроенности струн инструмента.

В качестве основного вычислительного элемента устройства будет использована плата Arduino UNO. Выбор данной платы мотивирован тем, что она уже у меня была. Разработка программного обеспечения будет происходить в интегрированной среде разработки для Windows Arduino IDE 1.8.16. В данной среде есть все необходимое для написания программного обеспечения с последующей загрузки на плату.

Разработка курсового проекта будет происходить поэтапно. В первую очередь необходимо подобрать элементы устройства, учитывая их надежность, стоимость, функциональность и размеры. Затем необходимо собрать устройство и разработать программное обеспечение для корректной обработки информации и поддержания связи между элементами схемы.

1 ОБЗОР КОМПОНЕНТОВ

1.1 Состав устройства

Как сказано ранее, разрабатываемое микропроцессорное устройство выполняет функции анализа аналогового сигнала, а если быть точнее - измерения частоты аналогового сигнала и определения тона. Для решения этих задач в состав устройства должны входить:

- микроконтроллер
- усилитель
- гнездо для кабеля
- кабель
- дисплей для вывода информации
- питание

1.2 Микроконтроллеры

Существует огромное разнообразие плат с разными микроконтроллерами. Все они отличаются размерами, параметрами, предустановленными интерфейсами и выполняемыми задачами. Для сравнения была выбрана плата Arduino UNO и аналоги других производителей. Результаты сравнения приведены в таблице 1.1.

Таблица 1.1 — Сравнение микроконтроллеров

Параметры сравнения	**Arduino UNO**	**Raspberry Pi Zero**	**NodeMcu Lua**
Микроконтроллер	ATmega328	ARM Cortex-A53	ESP8266
Входное напряжение	6 – 20 В	6 – 28 В	3.6 – 20 В
Флэш-память	32 Кб	порт для microCD	4 мб
ОЗУ	2 Кб	512 Мб	20 Кб
Тактовая частота	16 МГц	1 ГГц	80-160 МГц
Разрядность	8 бит	32 бит	32 бит
Цифровые входы/выходы	14 шт	26 шт	11 шт
Аналоговые входы/выходы	6 шт	0 шт	1 шт
Выходное напряжение	3.3В, 5 В	3.3В, 5 В	3.3В, 5 В
Рабочая температура	от -25 до +85 ºС	от -40 до +85 ºС	от -25 до +85 ºС

Для получения более подробной информации о рассмотренных микроконтроллерах использовались источники [4, 5, 6].

1.3 Операционные усилители

Сам по себе сигнал, исходящий из гитары, очень слабый и для анализа требует усиления.

Для данной схемы выбран операционный усилитель TL082, но существует куда больше усилителей, выполняющих разные задачи. Результаты сравнения приведены в таблице 1.2.

Таблица 1.2 — Сравнение усилителей

Параметры сравнения	**TL082**	**OPA2134**	**LT1803**
Коэффициент усиления	86	104-120	50
Полоса пропускания	5 MHz	8 MHz	85 MHz
Рабочая температура	от -40 до +125 ºC	от -55 до +125 ºC	от -40 до +85 ºC
Кол-во каналов	2	2	1

Для получения точной информации о данных датчиках использовалась техническая спецификация [8, 9, 10].

1.4 Гнездо и кабель

Почти все профессиональное музыкальное оборудование взаимодействует между собой посредством кабелей Jack 6.3мм. Для устройства представленного в данной курсовой работе был выбран именно такой формат кабеля и гнезда. Существует два типа кабелей Jack 6.3: mono и stereo. В данной курсовой работе этот параметр не имеет значения так как анализируется один канал сигнала.

1.5 Дисплей

Для того чтобы устройством было комфортно пользоваться и снимать данные в схеме предусмотрен небольшой OLED дисплей с разрешением 128x64. OLED дисплей не требует для себя отдельной схемы питания. Его питание обеспечивается выходами GND и 5V микроконтроллера

Сигнал, обработанный программой в символы/числовые значения, будет передан на дисплей с выходов микроконтроллера A4 и A5

Дисплей поддерживает желтый и синий цвета.

1.6 Питание

Для работы устройства требуется питание. В данной схеме использовались две батарейки 6F22 (“Крона”). Модуль усиления требует для себя питание равное 18В, в том время как допустимое питание микроконтроллера колеблется от 7 до 12В. Для питания модуля усиления два элемента питания были соеденены параллельно. А для питания микроконтроллера Arduino Uno питание было отведено от одного элемента питания. Таким образом входное напряжение микроконтроллера равное 9В является приемлемым для работы данного устройства.

2 РАЗРАБОТКА СТРУКТУРЫ УСТРОЙСТВА

2.1 Постановка задачи

Для того, чтобы составить структуру разрабатываемого устройства, необходимо выделить функции, которые будет выполнять устройство, затем определить компоненты и связь между ними исходя из данных функций. Результаты можно посмотреть на структурной схеме, представленной в приложении А.

В рамках данного курсового проекта необходимо разработать анализатор звукового сигнала. Для реализации было выбрано устройство, определяющее частоту сигнала (тон), исходящего от электрогитары и вывод этой информации на дисплей с выходов микроконтроллера. Вывод на дисплей должен отображать отклонение от ноты для точной настройки инструмента с погрешностью в 3Гц.

2.2 Определение компонентов структуры устройства

Компоненты структуры устройства выбираются исходя из функций, определенных в постановке задачи. Проанализировав выделенные функции, были определены следующие компоненты, представленные ниже.

1) Микроконтроллер — ключевой компонент всей схемы. Выполняет функцию обработки поступающей информации и выдает значение.

2) Питание — источние постоянного напряжения 9V.

3) Модуль усиления сигнала – компонент схемы, усиливающий слабый аналаговый сигнал для дальнейшего анализа микроконтроллером.

4) Модуль отображения – компонент схемы, представляющий результат анализа звукового сигнала в понятной человеку форме (в виде цифр/букв/символов).

2.3 Взаимодействие компонентов устройства

При колебании струн электрогитары, электромагнитный сигнал поступает на вход модуля усиления по кабелю Jack 6.3. Усиленный сигнал с выхода модуля усиления анализируется микроконтроллером и значение частоты исходного сигнала выводятся на OLED дисплей

Модуль питания взаимодействует со всеми элементами схемы напрямую или через модуль усиления, благодаря ему осуществляется питание всех необходимых элементов.

3 ОБОСНОВАНИЕ ВЫБОРА УЗЛОВ, ЭЛЕМЕНТОВ ФУНКЦИОНАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

3.1 Обоснование выбора микроконтроллеров

В данной курсовом проекте в качестве контроллера могла быть использована любая плата из представленных в таблице 1.1 так как устройство не требует больших затрат в памяти и мощности для корректной работы.

Контроллер Raspberry PI Zero является чем-то большим чем просто микроконтроллером. Данный “мини-компьютер” имеет процессор на 1ГГц и 512мб оперативной памяти что в сотни раз больше чем аналогичные характеристики других микроконтроллеров. И именно по этой причине Raspberry PI Zero не подходит для решения относительно простой задачи анализа звукового сигнала и его пераметры будут излишни.

NodeMcu Lua с другой стороны самый близкий к Arduino контроллер, со встроенным WI-FI модулем. Он превосходит Arduino Uno по основным характеристикам и в разы меньше по размерам. Но для разработки полу-аналаговой схемы работать с таким контроллем не комфортно из-за малых габаритов и отсутствия отдельного входа питания, который в Arduino Uno присутсвует.

В связи со всеми вышеперечисленными причинами, был выбран контроллер Arduino Uno который уже имелся в наличии.

3.2 Обоснование выбора операционного усилителя

Операционные усилители семейства TL являются самыми популярными и доступными операционными усилителями. Стоимость TL082 в несколько раз меньше стоимости OPA2134 при почти одинаковых характеристиках, а усилитель LT1803 хоть и обладает более лучшими характеристиками, тем не менее почти не встречается в продаже.

3.3 Обоснование выбора кабеля и гнезда

Для данных компонентов почти не существует альтернатива. Электрогитара имеет выход Jack 6.3-mono. Кабель расчитан на прямое подключение к гитаре без использования переходников которые сильно влияют на шум сигнала. По таким соображение в качестве входной точки схемы было выбрано гнездно польностью сочетающиеся с кабелем – Jack 6.3-mono. Это стандартное гнездо, используемое во всех музыкальных приборах начиная от тюнеров и комбо-усилителей, заканчивая профессиональным студийным оборудованием, колонками и наушниками.

3.4 Обоснование выбора дисплея

В данной курсовой работе предусмотрен вывод результата анализа звукового сигнала на дисплей в качестве цифр, буква и символов. Для данной задачи был выбран OLED дисплей небольшого разрешения 128x64 пикселя. Стоимость OLED дисплея в несколько раз превышает стоимость ЖК-дисплея с аналогичным разрешением из-за различных типов матриц и способа передачи изображения. Но OLED дисплей обладает лучшими характеристиками цветопередачи и отзывчивости, а также прост в использовании.

3.5 Обоснование выбора питания

Для данной схемы предусмотрено питание от двух батареек 6F22 (“Крона”) отдельно для микроконтроллера и схемы с усилителем. При использовании батареек прибор становиться мобильным и появляется возможность его использования без подключения к сети постоянного напряжения. Микроконтроллер Arduino UNO требует для себя питания 9V. Поэтому были выбраны батарейки типа 6F22 (“Крона”), обеспечивающие данное напряжение.

4 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ УСТРОЙСТВА

4.1 Расчёт мощности элементов схемы

Потребляемая мощность разрабатываемого устройства равна сумме мощностей, потребляемых его элементами. Расчет мощности элементов схемы устройства управления и самого устройства представлены в таблице 4.1 и 4.2 соответственно.

Таблица 4.1 – Расчет мощности элементов схемы устройства

Блок	U, В	I, мА	Кол-во	P, мВт
Микроконтроллер Arduino UNO R3	5	22	1	110
Модуль усиления	18	28	1	140
Модуль отображения	5	16	1	80
Суммарная мощность, мВт				430

В реализованной схеме главными компонентами являются: микроконтроллер Arduino UNO R3, модуль усиления на базе операционного усилителя TL082, OLED дисплей 0.96 с разрешением 128x64

Таким образом потребляемая мощность будет равна:

$P = 5 \cdot 22 + 18 \cdot 28 + 5 \cdot 16 = 430$ мВт.

Учитывая поправочный коэффициент в 20%, максимальная потребляемая мощность составит 516 мВт.

Рассчитаем потребляемый ток:

$$I = \frac{P}{U} = \frac{0.430}{5} = 0.086 \approx 0.09 \text{ А}$$

4.2 Микроконтроллер

Информация о выбранном микроконтроллере Arduino UNO представлена в пункте 3.1 раздела 3.

Микроконтроллер соединен напрямую только с выходом модуля усиления. В схеме с устройством к аналоговому входу A0 подключен выход с модуля усиления, который представляет собой аналоговый сигнал для анализа. Дисплей подключается к пинам GND, 5V, A4, A5 микроконтроллера

Данный микроконтроллер питается от напряжения 9 В.

5 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

5.1 Требования к разработке программного обеспечения

Разработанное микропроцессорное устройство анализирует аналаговый усиленный сигнал, исходящий от гитары. Под анализом сигнала в данном проекте подразумевается определение частоты сигнала, а затем определение ноты.

Устройство работает следующим образом. В линейный вход (гнездо Jack 6.3) вставляется шнур, который соединяет выход гитары с входом модуля усиления. На устройство подается питание путем подключения двух батареек типа “Крона”. Устройство готово к работе. При взаимодействии со струнами гитары, возникает звуковая волна, которая считывается магнитными звукоснимателя внутри гитары и звуковой сигнал передается далее по кабелю. Так как звуковой сигнал от гитары очень слыбый, ведь для работы электро-гитары не требуется питание, то для анализа этого сигнала его нужно усилить. Сигнал усиливается в схеме с операционным усилителем TL082 и уже усиленный сигнал подается на вход A0 микроконтроллера Arduino.

5.2 Блок-схема алгоритма

Блок-схема — это схематичное представление процесса, системы или компьютерного алгоритма. Блок-схемы часто применяются в разных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых логичных диаграмм.

Рассмотрим блок-схему алгоритма программного обеспечения данного курсового проекта, представленную в приложении Г.

На первом листе приложения представлена блок схема алгоритма самого устройства. Блоки 2 – 3 представляют собой подготовку программы для дальнейшей работы (инициализация переменных и определение модулей, подключенных к микроконтроллеру). Ключевыми являются блоки 5 – 29, которые реализуют саму логику программы в бесконечном цикле. В блоках 5, 9, 15 происходит получение данных с датчиков освещенности, расстояния и газа соответственно. В блоках 6 – 8, 11 – 14, 16 – 18 данная информация анализируется и, при необходимости, включается соответствующая индикация. В 20 блоке проверяется наличие входящего сообщения от радиопередатчика, полученного в 19 блоке. Если данные присутствуют, то, в зависимости от полученного кода (блоки 21 – 29) происходит движение, заданное пользователем.

На втором листе приложения представлена блок схема пульта дистанционного управления устройством. Блоки 2 – 3 аналогичны блокам устройства. Блоки 5 – 6 реализуют получение информации о действиях

пользователя, а блоки 7 – 15 анализируют полученные данные и выставляют соответствующий код на радиопередатчик. В блоке 16 информация отправляется устройству.

5.3 Исходный код программы для устройства управления

Пульт дистанционного управления считывает показания с двухосевых джойстиков, а затем преобразует данные для быстрой и удобной отправки по радиоканалу. Исходный код программного обеспечение под данное устройство можно найти в приложении Д (строки 1 – 53).

Функция `void setup()` (строки 22 – 27) необходима для начальной настройки контроллера, здесь задаются входные и выходные пины, а так же другие настройки. В данном случае `vw_setup(2000)` (строка 23) настраивает наш передатчик, а `pinMode(pinX, INPUT)` и `pinMode(pinY, INPUT)` (строки 25 и 26 соответственно) задают пины джойстика, как входные значения на плату.

Функция `void loop()` (строки 29 – 48) является главной, циклической функцией и работает на протяжении всей работы микроконтроллера. Здесь задается основная логика работы микроконтроллера. На начальном этапе работы функции обнуляются входные значения с джойстиков, а также очищается строка для отправки информации на радиоприёмник. Затем считываются новые данные с джойстиков, на основе полученных значений формируется новое сообщение, которое хранит номер операции для взаимодействия с устройством, подключенному к приёмнику. После этого сообщение конвертируется в массив `char` и передается на радиопередатчик для отправки.

Функция `void send (char *message)` (строки 50 – 53) получает в качестве параметра массив символов, которые необходимо отправить на радиоприёмник. Полученный массив отправляется и затем происходит ожидание полной отправки сообщения.

5.4 Исходный код программы для передвижного устройства

Данное устройство является основным в разрабатываемом курсовом проекте и здесь реализована большая часть логики. Исходный код программного обеспечение можно найти в приложении Д (строки 58 – 235).

Функция `void setup()` (строки 97 – 115) выполняет такие же задачи, как и в устройстве управления. В строках 98 – 99 настраивается радиоприемник, затем в строках 101 – 114 задаются входные и выходные пины устройства.

Функция `void loop()` (строки 117 – 124) является главной, здесь вызываются все методы для выполнения поставленных задач устройства. Она также работает на протяжении всей работы микроконтроллера.

Функция `void checkIlumination()` (строки 127 – 136) реализует логику взаимодействия с датчиком освещенности, здесь происходит считывание информации (строка 128) с последующей проверкой минимального допустимого значения. Строки 131 и 134 включают и выключают собственное освещение устройства соответственно.

Функция `void checkDistanceToObject()` (строки 138 – 159) реализует логику взаимодействия с датчиком расстояния. В строках 140 – 145 посылается два ультразвуковых сигнала равные 2 и 10 миллисекунд для получения информации о расстоянии до ближайшего объекта. В строке 149 полученное значение переводится в сантиметры, после чего (строки 151 – 158) сравнивается с минимально допустимым и, в зависимости от результата включаются и выключаются боковые светодиоды устройства (строки 153 и 157 соответственно), а также происходит остановка устройства, если значение довольно мало.

Функция `void checkOfGas()` (строки 161 – 171) реализует логику взаимодействия с датчиком горючих газов. Изначально происходит считывание информации с данного датчика (строки 162 – 163), а затем проверка допустимых значений (строки 165 – 169), если значение выше допустимого, то вызывается метод `turnOnSound()`, в ином случае метод `turnOffSound()` о которых будет сказано ниже.

Функция `void turnOnSound()` (строки 173 – 175) включает пьезодинамик, а функция `void turnOffSound()` (строки 177 – 179) его выключает.

Функция `void selectStateOfMotors(char state)` (строки 181 – 198) получает в качестве параметра состояние движения, которое отправил пользователь. В теле метода происходит сравнение со всеми возможными состояниями и, в зависимости от результатов сравнения, вызываются методы для работы с моторами.

Функции `void mooveForward()` (строки 200 – 205) , `void mooveBack()` (строки 207 – 212), `void turnLeft()` (строки 214 – 220) , `void turnRight()` (строки 222 – 228) и `void stand()` (строки 230 – 235) реализуют логику взаимодействия с моторами: движение прямо, назад, влево, вправо и бездействие соответственно.

ЗАКЛЮЧЕНИЕ

В результате работы над данным курсовым проектом было разработано работоспособное микропроцессорное устройство со своим программным обеспечением. Устройство анализирует аналоговый сигнал, передающийся от гитары на микроконтроллер, где осуществляется обработка этого сигнала и вывод информации на дисплей в виде ноты. Данный проект был спроектирован в соответствии с поставленными задачами, весь функционал был реализован в полном объеме.

Разработанное микропроцессорное устройство обладает следующими достоинствами: низкая стоимость компонентов, простота реализации и сборки. Недостатки данного устройства: сильная энергозависимость, необходимость подключения гитары через кабель.

В дальнейшем планируется усовершенствование данного курсового проекта. Одним из таких улучшений является оптимизация алгоритма анализа полученных данных, улучшение питания, а также создание более дружественного интерфейса.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1]. Вычислительные машины, системы и сети: дипломное проектирование (методическое пособие) [Электронный ресурс] : Минск БГУИР 2019. – Электронные данные. – Режим доступа: https://www.bsuir.by/m/12_100229_1_136308.pdf

[2]. Документация Arduino [Электронный ресурс]. – Электронные данные. – Режим доступа: https://docs.arduino.cc/

[3]. Геддес, М. 25 крутых проектов с Arduino / М. Геддес ; [пер. с англ. М. А. Райтмана]. – Москва : Эксмо, 2019. – 272 с.

[4]. Arduino UNO [Электронный ресурс]. – Электронные данные. – Режим доступа: http://arduino.ru/Hardware/ArduinoBoardUno – Дата доступа: 11.09.2023

[5]. TL082 Datasheet [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.alldatasheet.com/view.jsp?Searchword=Tl082%20datasheet – Дата доступа: 09.10.2023

[6]. OLED дисплеи [Электронный ресурс]. – Электронные данные. – Режим доступа: https://amperkot.by/page/amperkot-oled-displays/ – Дата доступа: 17.10.2023

[7]. Аналоговая микросхемотехника [Электронный ресурс]: Минск БГУИР 2003. – Электронные данные. – Режим доступа: https://www.bsuir.by/m/12_100229_1_85477.pdf – Дата доступа: 12.10.2023

[8]. Программирование Arduino [Электронный ресурс] - Электронные данные. – Режим доступа: https://arduino.ru/Reference – Дата доступа: 1.11.2023

[9]. Пайка своими руками: основы для начинающих - [Электронный ресурс] - Электронные данные. – Режим доступа: https://vopros-remont.ru/elektrika/pajka/ – Дата доступа: 4.11.2023

ПРИЛОЖЕНИЕ А
(обязательное)

Схема структурная

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема функциональная

ПРИЛОЖЕНИЕ В

(обязательное)

Схема принципиальная

ПРИЛОЖЕНИЕ Г
(обязательное)

Схема программы

ПРИЛОЖЕНИЕ Д
(обязательное)

Листинг кода

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

//clipping indicator variables
boolean clipping = 0;

//data storage variables
byte newData = 0;
byte prevData = 0;
unsigned int time = 0;//keeps time and sends vales to store in timer[]
occasionally
int timer[10];//storage for timing of events
int slope[10];//storage for slope of events
unsigned int totalTimer;//used to calculate period
unsigned int period;//storage for period of wave
byte index = 0;//current storage index
float frequency;//storage for frequency calculations
int maxSlope = 0;//used to calculate max slope as trigger point
int newSlope;//storage for incoming slope data

//variables for decided whether you have a match
byte noMatch = 0;//counts how many non-matches you've received to reset
variables if it's been too long
byte slopeTol = 3;//slope tolerance- adjust this if you need
int timerTol = 10;//timer tolerance- adjust this if you need

//variables for amp detection
unsigned int ampTimer = 0;
byte maxAmp = 0;
byte checkMaxAmp;
byte ampThreshold = 5;//raise if you have a very noisy signal

void setup(){
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  delay(500);

  display.setTextSize(4);
  display.setTextColor(WHITE);
  // Display static text

  Serial.begin(9600);
```

```
  pinMode(13,OUTPUT);//led indicator pin
  pinMode(12,OUTPUT);//output pin

  cli();//diable interrupts

  //set up continuous sampling of analog pin 0 at 38.5kHz

  //clear ADCSRA and ADCSRB registers
  ADCSRA = 0;
  ADCSRB = 0;

  ADMUX |= (1 << REFS0); //set reference voltage
  ADMUX |= (1 << ADLAR); //left align the ADC value- so we can read highest 8
bits from ADCH register only

  ADCSRA |= (1 << ADPS2) | (1 << ADPS0); //set ADC clock with 32 prescaler-
16mHz/32=500kHz
  ADCSRA |= (1 << ADATE); //enabble auto trigger
  ADCSRA |= (1 << ADIE); //enable interrupts when measurement complete
  ADCSRA |= (1 << ADEN); //enable ADC
  ADCSRA |= (1 << ADSC); //start ADC measurements

  sei();//enable interrupts
}

ISR(ADC_vect) {//when new ADC value ready

  PORTB &= B11101111;//set pin 12 low
  prevData = newData;//store previous value
  newData = ADCH;//get value from A0
  if (prevData < 127 && newData >=127){//if increasing and crossing midpoint
    newSlope = newData - prevData;//calculate slope
    if (abs(newSlope-maxSlope)<slopeTol){//if slopes are ==
      //record new data and reset time
      slope[index] = newSlope;
      timer[index] = time;
      time = 0;
      if (index == 0){//new max slope just reset
        PORTB |= B00010000;//set pin 12 high
        noMatch = 0;
        index++;//increment index
      }
      else if (abs(timer[0]-timer[index])<timerTol && abs(slope[0]-
newSlope)<slopeTol){//if timer duration and slopes match
        //sum timer values
        totalTimer = 0;
        for (byte i=0;i<index;i++){
          totalTimer+=timer[i];
        }
        period = totalTimer;//set period
        //reset new zero index values to compare with
        timer[0] = timer[index];
        slope[0] = slope[index];
        index = 1;//set index to 1
        PORTB |= B00010000;//set pin 12 high
        noMatch = 0;
      }
```

```
      else{//crossing midpoint but not match
        index++;//increment index
        if (index > 9){
          reset();
        }
      }
    }
    else if (newSlope>maxSlope){//if new slope is much larger than max slope
      maxSlope = newSlope;
      time = 0;//reset clock
      noMatch = 0;
      index = 0;//reset index
    }
    else{//slope not steep enough
      noMatch++;//increment no match counter
      if (noMatch>9){
        reset();
      }
    }
  }

  if (newData == 0 || newData == 1023){//if clipping
    clipping = 1;//currently clipping
    //Serial.println("clipping");
  }

  time++;//increment timer at rate of 38.5kHz

  ampTimer++;//increment amplitude timer

  if (abs(127-ADCH)>maxAmp){
    maxAmp = abs(127-ADCH);
  }

  if (ampTimer==1000){
    ampTimer = 0;
    checkMaxAmp = maxAmp;
    maxAmp = 0;
  }
}

void reset(){//clean out some variables
  index = 0;//reset index
  noMatch = 0;//reset match couner
  maxSlope = 0;//reset slope
}

void checkClipping(){//manage clipping indication
  if (clipping){//if currently clipping
    clipping = 0;
  }
}

double openStringFrequencies[] = {82.41, 110.00, 146.83, 196.00, 246.94,
329.63};
String stringNames[] = {"E", "A", "D", "G", "B", "E"};

bool in_between(double a, double min, double max)
```

```
{
   return a >= min && a <= max;
}

String semi_ton(String note)
{
  String semiton;

  if(note == "E") {
    semiton = "F";
  }
  else if(note == "B") {
    semiton = "C";
  }
  else {
    semiton = note + "#";
  }

  return semiton;
}

String note = "";

String getGuitarNoteByFrequency(double frequency) {
    double accuracyTone = 1;
    double accuracyDemiton = 3;

    // Determine the note based on the provided frequency.
    for (int i = 0; i < 6; i++) {
      if(in_between(frequency, openStringFrequencies[i] - accuracyTone,
openStringFrequencies[i] + accuracyTone))
        note = stringNames[i];
      else if (in_between(frequency, openStringFrequencies[i] + accuracyTone,
openStringFrequencies[i] + accuracyDemiton))
        note = stringNames[i] + "+";
      else if (in_between(frequency, openStringFrequencies[i] +
accuracyDemiton, openStringFrequencies[i] + 2*accuracyDemiton))
        note = semi_ton(stringNames[i]);
      else if (in_between(frequency, openStringFrequencies[i] - accuracyTone
- accuracyDemiton, openStringFrequencies[i] - accuracyTone))
        note = stringNames[i] + "-";
    }

    return note;
}

void printNote(float frequency) {
    String note = getGuitarNoteByFrequency(frequency);
    display.setCursor(60, 20);
    display.clearDisplay();
    display.print(note);
    display.display();
}

void loop(){
  checkClipping();

  if (checkMaxAmp > ampThreshold){
```

```
    frequency = 38462/float(period);//calculate frequency timer rate/period

    //print results
    Serial.print(frequency);
    Serial.println(" hz");
    printNote(frequency);
  }

  delay(150);
}
```

ПРИЛОЖЕНИЕ Е
(обязательное)

Перечень элементов

ПРИЛОЖЕНИЕ Ж
(обязательное)

Ведомость документов