МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доц., канд. техн. наук		Т.Н.Соловьёва
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

ГЕНЕРАТОР СИНУСОИДАЛЬНЫХ СИГНАЛОВ

по дисциплине: МИКРОКОНТРОЛЛЕРНЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4143		Е.Д.Тегай
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

студенту группы 4143 (номер) Тегай Екатерине Дмитриевне (фамилия, имя, отчество)

на тему Генератор синусоидальных сигналов

Цель проекта: разработка аппаратно-программного комплекса генератора синусоидальных сигналов на базе микроконтроллера серии 8051 в системе автоматизированного проектирования Proteus.

Задачи, подлежащие решению: реализация возможности задания частоты, фазы и амплитуды синусоидального сигнала,
генерация синусоидального напряжения заданной амплитуды, фазы и частоты.

Содержание пояснительной записки (основные разделы):
проектирование аппаратного обеспечения,
проектирование программного обеспечения,
тестирование и отладка аппаратно-программного комплекса.

Срок сдачи работы « 30 » ноября 2024

Руководитель

доц., канд. техн. наук		Т.Н. Соловьева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Задание принял к исполнению

студент группы № 4143		Е.Д.Тегай
	подпись, дата	инициалы, фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Генерация синусоидальных сигналов играет важную роль в радиотехнических, измерительных и управляющих системах. Синусоидальный сигнал является одним из основных видов сигналов в электронике и связи, используемых для тестирования и калибровки оборудования, моделирования различных процессов, а также в медицинских, музыкальных и научных приборах. Генератор, в котором можно менять параметры синусоидального сигнала, а именно амплитуду, частоту и фазу, представляет собой универсальный инструмент, который способен адаптировать параметры сигнала для множества целей, что делает его особенно полезным в процессе разработки и тестирования электронных схем и устройств.

Актуальность данной темы обусловлена потребностью в настраиваемых генераторах сигналов для лабораторного и практического применения. Применение микроконтроллерных технологий позволяет создавать гибкие и легко настраиваемые генераторы, обеспечивающие простоту управления и настройку параметров сигнала. Современные микроконтроллеры, например серии 8051, предоставляют широкие возможности для интеграции функций генерации и обработки сигналов, а также обеспечения взаимодействия с другими устройствами через интерфейсы, что делает разработку подобных систем доступной и эффективной.

Целью данной курсовой работы является разработка генератора синусоидальных сигналов с возможностью изменять амплитуду, частоту и фазу сигнала. Устройство моделируется и тестируется в среде Proteus, что позволяет увидеть результат работы устройства в реальном времени и скорректировать возможные ошибки на этапе разработки.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Обосновать выбор компонентов для схемы генератора.

2. Разработать схему генератора синусоидального сигнала с учётом возможности изменения амплитуды, частоты и фазы.

3. Написать программное обеспечение для управления генерацией сигнала и взаимодействия с клавиатурой, позволяющее задавать необходимые параметры.

4. Моделировать работу устройства в среде Proteus, оценить его работоспособность и корректность изменения параметров.

5. Провести анализ полученных данных, выявить преимущества и возможные недостатки устройства, а также перспективы его улучшения.

В результате выполнения работы будет разработан функциональный генератор синусоидальных сигналов, который может быть использован для тестирования и калибровки различных устройств, что подтверждает его практическую значимость и полезность для широкого спектра задач.

1. ПРОЕКТИРОВАНИЕ АППАРАТНОГО ОБЕСПЕЧЕНИЯ

В данном разделе описан процесс проектирования аппаратной части генератора синусоидальных сигналов. Аппаратная реализация включает микроконтроллер серии 8051, клавиатуру для ввода параметров, цифро-аналоговый преобразователь (ЦАП), световые индикаторы для каждого из параметров и осциллограф для визуализации выходного сигнала.

Проектирование схемы и моделирование работы устройства выполнены в среде Proteus, что позволило визуализировать работу системы. Далее будут рассмотрены обзор существующих решений, обоснование выбора аппаратной реализации, разработка схемы и характеристика компонентов, использованных в проекте.

1.1. Обзор существующих решений

Генераторы синусоидальных сигналов широко используются в радиоэлектронике, телекоммуникациях и системах автоматизации для создания сигналов различной частоты и амплитуды. Основные типы таких генераторов включают аналоговые генераторы, генераторы на основе микроконтроллеров и специализированные интегральные схемы (IC). Рассмотрим подробнее каждый из них.

1. *Аналоговые генераторы*. Аналоговые генераторы синусоидальных сигналов обычно строятся на основе RC- или LC-цепей, которые создают устойчивое колебание определенной частоты. В таких устройствах генерация сигнала достигается путём создания положительной обратной связи, которая поддерживает гармоническое колебание. Преимущества аналоговых генераторов включают простоту схемы и плавность выходного сигнала. Однако они имеют ограниченные возможности для регулировки частоты и амплитуды, а также обычно требуют ручной настройки параметров, что снижает их гибкость и универсальность. Эти устройства хорошо подходят для простых задач, но недостаточно удобны для приложений, где требуется автоматическое или цифровое управление параметрами. Пример схемы RC-генератора синусоидальных сигналов продемонстрирован на рисунке 1.

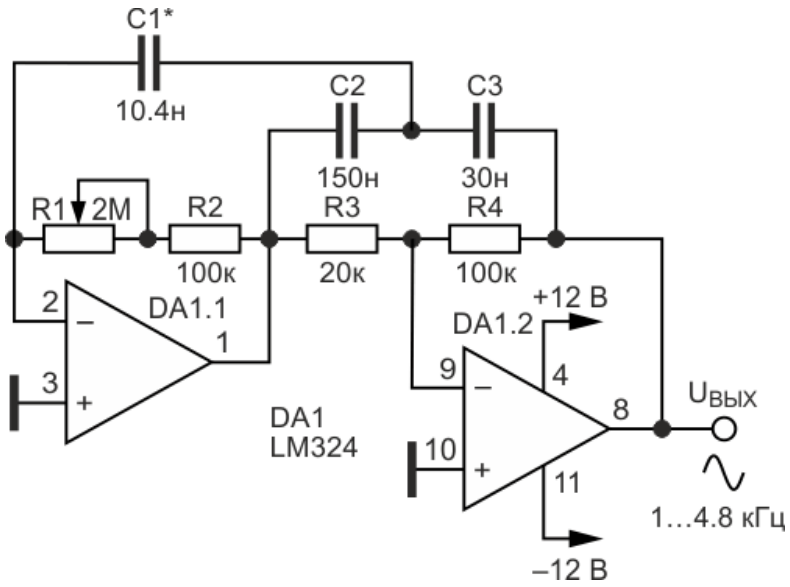

Рисунок 1 – RC-генератор синусоидальных сигналов

2. *Генераторы на основе микроконтроллеров*. Генераторы на базе микроконтроллеров (например, Arduino, микроконтроллеры серии 8051 и другие) стали популярными благодаря их гибкости и лёгкости программирования. С помощью программного обеспечения микроконтроллер может генерировать сигнал, а пользователь может менять параметры сигнала (амплитуду, частоту, фазу) с помощью кода или периферийных устройств, таких как клавиатура. Эти генераторы более универсальны по сравнению с аналоговыми, поскольку позволяют регулировать параметры в реальном времени, использовать различные формы сигналов и даже выводить цифровые сигналы на других устройствах. Основные недостатки генераторов на микроконтроллерах включают ограниченное разрешение и скорость изменения параметров, а также сложность в создании высокочастотных сигналов по сравнению с аналоговыми генераторами. Данный тип, собственно, и реализован в данном курсовом проекте. Пример такого генератора на основе микроконтроллера серии Arduino продемонстрирован на рисунке 2.

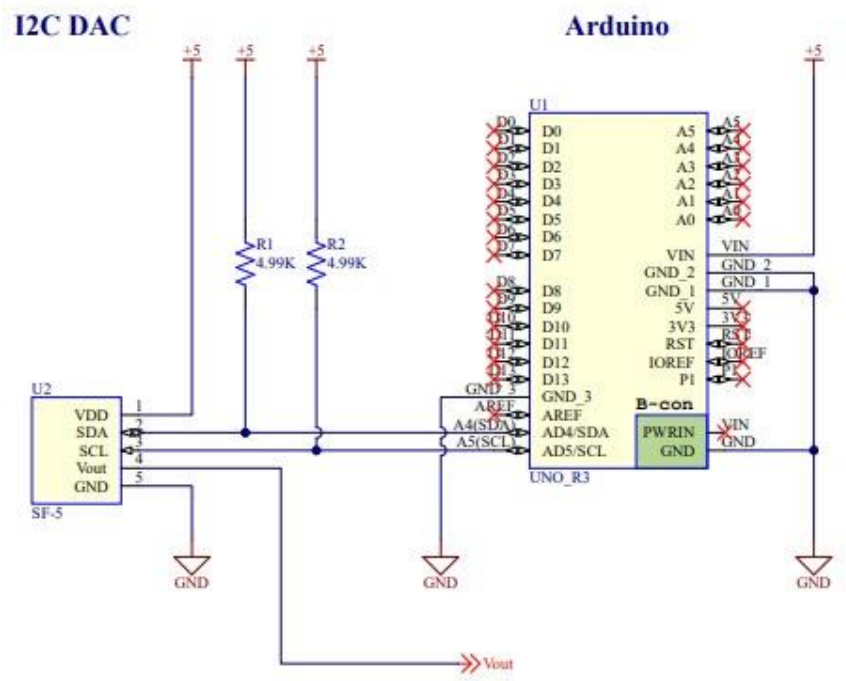

Рисунок 2 – Генератор на основе микроконтроллера серии Arduino

3. Интегральные генераторы (специализированные ИС). Генераторы синусоидальных сигналов на специализированных интегральных схемах (например, AD9833, MAX038) предоставляют высокую точность и стабильность параметров сигнала, и часто имеют встроенные цифровые интерфейсы для управления через микроконтроллеры. Эти интегральные схемы могут генерировать сигналы в широком диапазоне частот и обеспечивают высокую точность настройки параметров, но обычно стоят дороже и требуют подключения контроллера для изменения параметров. Основной недостаток интегральных генераторов — ограниченность по частоте или амплитуде, а также высокая стоимость по сравнению с аналоговыми или микроконтроллерными решениями. Пример такого генератора показан на рисунке 3.

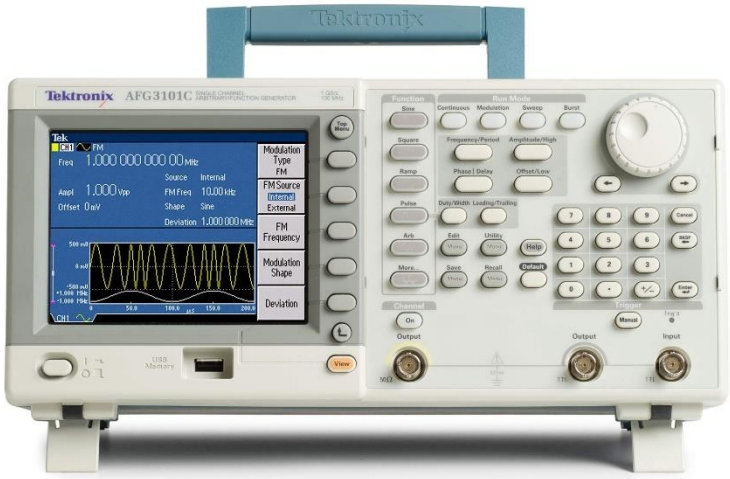

Рисунок 3 – Генератор синусоидального сигнала Tektronix AFG3101

Проанализировав существующие типы генераторов, можно отметить, что аналоговые генераторы имеют ограниченные возможности регулировки параметров, специализированные интегральные схемы могут оказаться дорогими для данного проекта, а микроконтроллерные генераторы обладают необходимой гибкостью и управляемостью.

1.2. Выбор аппаратной реализации

Для создания генератора в проекте используется микроконтроллер серии 8051 AT9C51, клавиатура для ввода параметров (KEYPAD-PHONE), цифро-аналоговый преобразователь LTC1450, осциллограф для наблюдения за сигналом и диагностические светодиоды с резисторами (LED-RED, LED-GREEN, LED-BLUE). Рассмотрим подробнее каждый из компонентов.

1. *Микроконтроллер*. Микроконтроллер AT89C51 является представителем серии микроконтроллеров 8051, который включает 8-разрядный процессор, 4 КБ встроенной памяти программ, 128 байт оперативной памяти и 32 порта ввода-вывода. Этот микроконтроллер поддерживает тактовую частоту до 24 МГц, что достаточно для работы в большинстве систем управления и генерации сигналов средней частоты.

Данный микроконтроллер имеет 32 порта ввода-вывода, которые можно использовать для подключения периферийных устройств. Рассмотрим подробнее каждый из портов ввода-вывода.

a) *XTAL1 и XTAL2*. Эти два вывода используются для подключения внешнего кварцевого резонатора или генератора, который задает тактовую частоту микроконтроллера. Первый является входным, второй, соответственно, выходным. Тактовая частота определяет скорость работы микроконтроллера. Для этого микроконтроллера обычно используют кварцевые резонаторы с частотой от 1 до 24 МГц.

b) *RST.* Этот вход используется для сброса микроконтроллера. При подаче высокого уровня (логической 1) на RST на время не менее двух машинных циклов микроконтроллер возвращается в начальное состояние. Все регистры и память инициализируются, и выполнение программы начинается с начала.

c) $\overline{PSEN}$*(Program Store Enable).* Это сигнал, который микроконтроллер использует для считывания данных из внешней памяти программ, если она подключена. Этот вход активен, когда микроконтроллер выполняет чтение программы из внешней памяти. Если используется внутренняя память программ, PSEN остается неактивным.

d) *ALE (Address Latch Enable).* Используется для мультиплексирования адреса и данных на порту P0. Он подает управляющий сигнал на внешние устройства, синхронизируя выбор адреса. Это позволяет совместно использовать один порт для адреса и данных.

e) $\overline{EA}$ (*External Access).* Данный вход определяет, будет ли микроконтроллер работать с внутренней или внешней памятью программ. Если EA подключен к 5 В, микроконтроллер использует внутреннюю память. Если EA подключен к земле, программа считывается из внешней памяти.

f) *P1.0 – P1.7.* Порт 1 — это 8-разрядный двунаправленный порт, каждый из выводов которого можно настроить как вход или выход. Он не является мультиплексированным, то есть все его линии используются

исключительно для ввода-вывода. Это позволяет легко управлять отдельными устройствами, подключенными к этим выводам.

g) *P0.0/AD0 – P0.1/AD7.* Порт 0 является двунаправленным и мультиплексированным. Он может использоваться как порт ввода-вывода, а также для передачи адреса/данных в режиме расширенной памяти. В режиме расширенной памяти выводы P0 используются для передачи младшего байта адреса (AD0-AD7) или данных. При работе с внешней памятью данные и младший байт адреса передаются через порт 0. Это делает порт 0 универсальным, хотя и требует управления мультиплексированием.

h) *P2.0/A8 – P2.7/A15.* Порт 2 также является двунаправленным, и в режиме расширенной памяти он используется для передачи старшего байта адреса (A8-A15). Если микроконтроллер работает с внутренней памятью, выводы порта 2 можно использовать как обычные линии ввода-вывода. Этот порт полезен для адресации, когда объем внешней памяти превышает 256 байт.

i) *P3.0/RXD (Receive Data)* и *P3.1/TXD (Transmit Data).* Первый вывод используется для приёма данных в последовательном режиме передачи, второй - ля передачи данных. Оба формируют интерфейс UART (универсальный асинхронный приемопередатчик), позволяя микроконтроллеру обмениваться данными с другими устройствами по последовательному интерфейсу.

j) *P3.2/$\overline{INT0}$ (External Interrupt 0)* и *P3.3/$\overline{INT1}$ (External Interrupt 1).* Эти два вывода служат для приема внешних прерываний. Прерывания используются для немедленной реакции микроконтроллера на внешние события, такие как сигнал от сенсора или устройства. INT0 и INT1 могут быть настроены на срабатывание по уровню или по фронту.

k) *P3.4/T0 (Timer 0 Input)* и *P3.5/T1 (Timer 1 Input).* Эти выводы используются для работы с таймерами. Таймеры используются для отсч`та времени или подсчета импульсов. Входы T0 и T1 позволяют использовать таймеры для измерения внешних событий или частоты сигналов.

l) *P3.6/$\overline{WR}$ (Write)* и *P3.7/$\overline{RD}$ (Read).* Первый является входом записи, второй – входом чтения. Оба активны при работе с внешней памятью данных. Применяются, если микроконтроллер взаимодействует с внешней памятью данных или устройствами, которые требуют чтения/записи.

Преимуществами являются компактность, низкая стоимость и достаточная производительность для управления генерацией синусоидального сигнала. Однако у него ограниченный объём памяти и низкая тактовая частота по сравнению с более современными микроконтроллерами, что может ограничивать частотный диапазон генератора.

Микроконтроллер является основополагающей проекта. Он управляет генерацией синусоидального сигнала и обеспечивает изменение параметров (амплитуды, частоты и фазы) на основе данных, вводимых с клавиатуры. Он также контролирует состояние светодиодов, индицирующих текущий режим настройки генератора.

Внешний вид микроконтроллера продемонстрирован на рисунке 4.

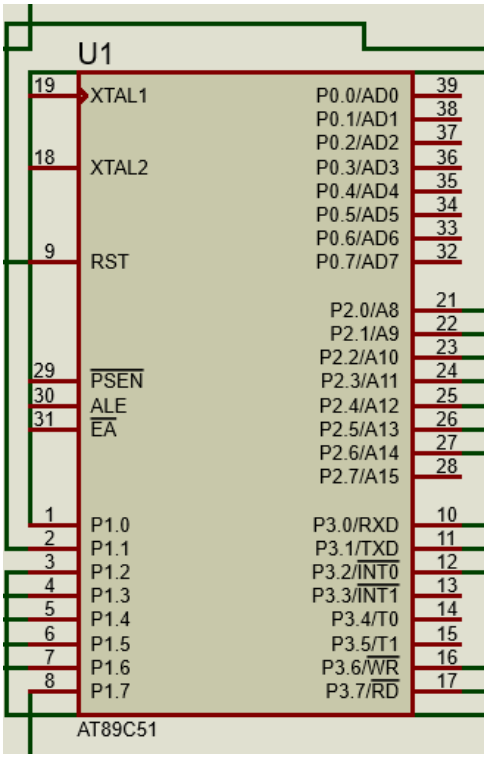

Рисунок 4 – Микроконтроллер

2. *Клавиатура.* Keypad-phone в Proteus представляет собой обычную цифровую клавиатуру с цифрами 0 – 9 и кнопками «*» и «#». Используется для ввода числовых данных и управления функциями устройства.

Клавиатура имеет несколько линий для строк и столбцов, которые подключаются к портам микроконтроллера. При нажатии определённой кнопки замыкается соответствующий контакт, что позволяет микроконтроллеру определить, какая кнопка была нажата.

Преимущества данной клавиатуры включают простоту использования и лёгкость интеграции с микроконтроллером. Недостатком является ограниченное количество кнопок, что может затруднить реализацию более сложных интерфейсов.

В проекте клавиатура служит для ввода параметров генератора: с её помощью можно задавать амплитуду, частоту и фазу сигнала, что делает устройство интерактивным и легко настраиваемым для пользователя.

Внешний вид клавиатуры показан на рисунке 5.

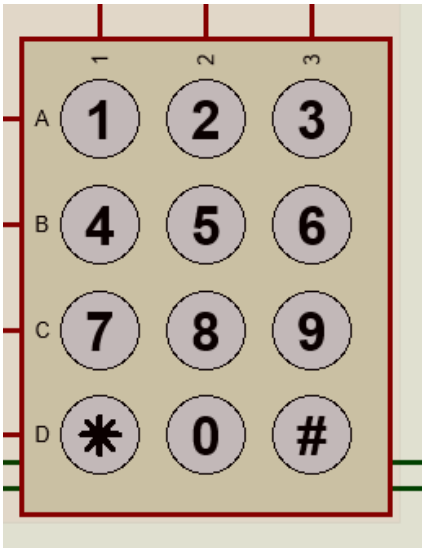

Рисунок 5 – Клавиатура

3. *Цифро-аналоговый преобразователь (ЦАП).* LTC1450 — это 12-битный цифро-аналоговый преобразователь, который преобразует цифровые сигналы из микроконтроллера в аналоговые, что необходимо для генерации синусоидального сигнала с плавной формой. LTC1450 поддерживает

напряжение питания от 2.7 до 5.5 В и имеет встроенные функции для точного вывода аналогового сигнала. Рассмотрим более подробно его входы и выходы.

1. *D0 – D11.* Это 12-битная шина данных, по которой передаются цифровые данные на вход ЦАП. D0 — младший значащий бит (LSB), а D11 — старший значащий бит (MSB). Эта шина позволяет передать 12-битное значение, которое затем преобразуется в аналоговый выходной сигнал на VOUT.

2. $\overline{WR}$. Это активный низкий сигнал, который используется для записи данных в ЦАП. Когда он устанавливается в низкий уровень, то происходит считывание данных с шины D0-D11, и ЦАП начинает преобразование. После снятия сигнала в высокий уровень ЦАП фиксирует значение.

3. $\overline{CSLSB}$ и $\overline{CSMSB}$ *(Chip Select).* Эти сигналы оба активны по низкому уровню и используются для выбора активации определенных частей ЦАП. Первый включает использование младших битов данных, тогда как второй – старших битов данных. Эти сигналы позволяют выбирать, какие биты данных будут участвовать в преобразовании.

4. $\overline{CLR}$. Этот сигнал служит для сброса всех данных, загруженных в регистры ЦАП. При подаче низкого уровня на этот вход все внутренние регистры устанавливаются в нулевое состояние, что приводит к выводу на VOUT нулевого напряжения.

5. $\overline{LDAC}$. Это активный низкий сигнал, используемый для загрузки значений из регистра данных в сам ЦАП. Когда он устанавливается в низкий уровень, данные из внутреннего регистра передаются в ЦАП, и выход VOUT обновляется. Этот вход позволяет управлять моментом, когда данные применяются к выходу.

6. *VOUT.* Это основной аналоговый выход, на котором формируется аналоговый сигнал на основе цифрового значения, поступившего на входную шину D0-D11. VOUT представляет собой результат преобразования и может

быть подключён к различным аналоговым устройствам, таким как осциллограф или аналоговые усилители.

7. REFOUT. Это выходное опорное напряжение, которое генерируется внутри ЦАП и может использоваться для стабилизации работы или как опорное напряжение для других компонентов схемы. Этот выходной сигнал может применяться в цепи, требующей точного и стабильного опорного напряжения.

8. REFHI и *REFLO.* Это выводы для подключения внешнего опорного напряжения. Первый является выводом для подключения верхнего опорного напряжения (например, +5 В), тогда как второй – нижнего опорного напряжения (заземление). Эти два вывода позволяют регулировать диапазон аналогового выходного сигнала VOUT в зависимости от требуемого диапазона напряжений.

9. X1/X2. Эти выводы предназначены для подключения внешних конденсаторов, чтобы стабилизировать работу схемы или уменьшить помехи. Иногда их используют для дополнительной фильтрации аналогового сигнала или для компенсации частотных отклонений.

Преимущества LTC1450 включают высокую точность (12-битное разрешение) и лёгкость в интеграции с микроконтроллерами. Недостатком может быть ограничение по выходному напряжению и частоте, которое влияет на диапазон генерации синусоидального сигнала.

В проекте ЦАП преобразует цифровые данные от микроконтроллера в аналоговый синусоидальный сигнал, который затем отображается на осциллографе. Этот компонент является ключевым для получения плавного и точного аналогового сигнала.

Внешний вид данного ЦАП показан на рисунке 6.

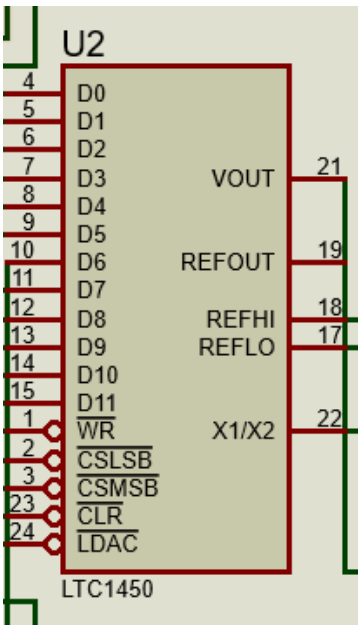

Рисунок 6 – ЦАП

4. *Осциллограф*. Осциллограф используется для визуализации выходного синусоидального сигнала и контроля за его параметрами. Он подключается к выходу ЦАП, что позволяет наблюдать форму и параметры сигнала, генерируемого устройством.

Преимущество осциллографа заключается в его наглядности и возможности точного наблюдения параметров сигнала. Недостатком является необходимость ручной настройки для конкретного диапазона частот и напряжений.

В проекте осциллограф служит для мониторинга синусоидального сигнала, отображая его в реальном времени. Это позволяет оценить качество работы генератора и проверить корректность изменений амплитуды, частоты и фазы.

Внешний вид осциллографа показан на рисунке 7.

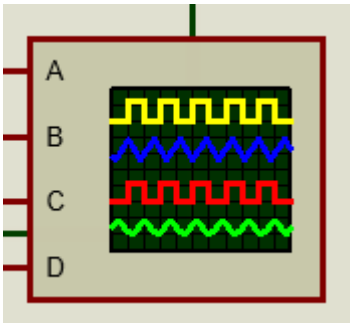

Рисунок 7 – Осциллограф

5. *Светодиоды и резисторы*. Светодиоды красного, зелёного и синего цвета используются для индикации режимов работы устройства. LED-RED отвечает за режим амплитуды, LED-GREEN — за режим частоты, а LED-BLUE — за режим фазы. Каждый светодиод подключён через резистор 0.1 Ом для ограничения тока и защиты от перегрузки. Не участвуют в логике работы вывода сигнала, а лишь дают понять пользователю, что в конкретный момент времени он меняет из параметров.

Каждый светодиод подключен к порту микроконтроллера, который управляет их включением и выключением в зависимости от текущего режима настройки.

Преимущества светодиодов включают низкое потребление энергии и простоту в использовании. Недостаток — ограниченные возможности индикации, так как они показывают только активность определенного режима без дополнительных деталей.

Внешний вид всех трёх светодиодов показан на рисунке 8.

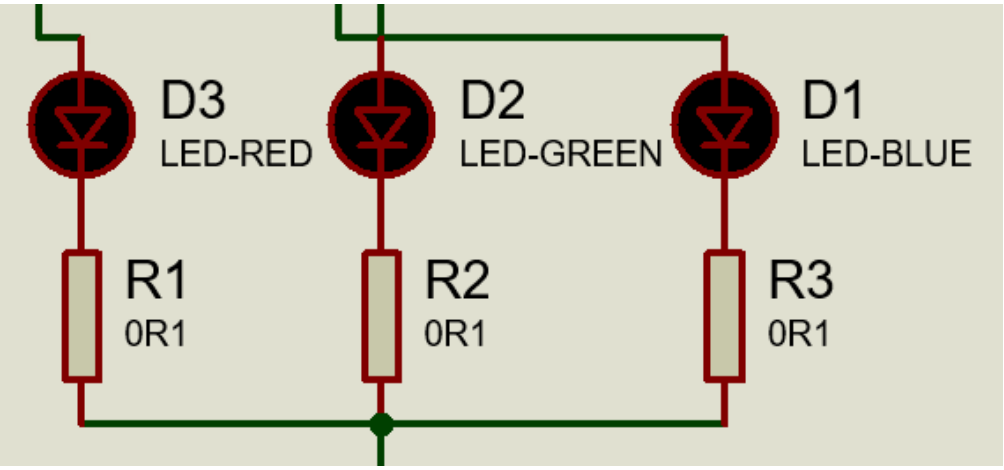

Рисунок 8 – Светодиоды

6. *Батарея.* Батарея на 5 В обеспечивает питание всех компонентов устройства, поддерживая стабильную работу схемы. Это напряжение подходит для микроконтроллера и остальных компонентов, таких как ЦАП и светодиоды.

Преимущество — стандартное и стабильное напряжение, подходящее для большинства низковольтных компонентов. Недостаток — необходимость

в защите от перегрузок и кратковременных скачков напряжения, что требует дополнительных конденсаторов для фильтрации.

Батарея обеспечивает схему необходимым напряжением, поддерживая стабильную работу всех компонентов и точность выходного сигнала.

Внешний вид батареи показан на рисунке 9.

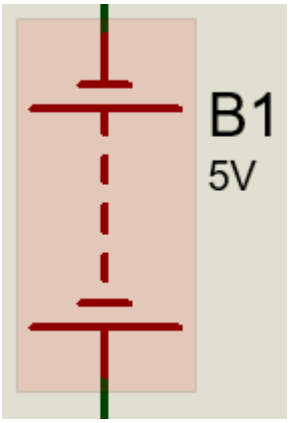

Рисунок 9 - Батарея

7. *Земля (GND) и питание (VCC/VDD).* Земля является общим проводом или «нулём», который служит заземлением для всех компонентов схемы. Весь ток в схеме возвращается через вывод GND к источнику питания, что замыкает электрическую цепь и делает ее рабочей. Кроме того, заземление служит общей точкой отсчета для всех напряжений в цепи, позволяя корректно определить амплитуду сигнала на выводе VOUT.

Питание отвечает за подачу основного питающего напряжения на микроконтроллер. Постоянное и стабильное напряжение питания на этом выводе важно для корректной работы всей схемы, поскольку от него зависит точность выходного сигнала микроконтроллера и устойчивость его работы.

Нестабильность на выводе питания может привести к появлению шумов и искажений на аналоговом выходе ЦАП, ухудшая точность сигнала. Неправильное подключение заземления может вызвать дрейф нуля, то есть смещение сигнала относительно точки отсчёта, что скажется на корректности работы всей системы.

Внешний вид питания и земли показан на рисунках 10 – 11 соответственно.

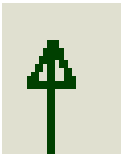

Рисунок 10 – Питание

Рисунок 11 - Земля

1.3. Описание разработки схемы проекта

Разработка схемы проекта начинается с непосредственно его создания в Proteus. Затем в схему добавляется главный её компонент – микроконтроллер. Первым делом к нему необходимо подключить питание. Было принято решение подключить его к порту RST. Затем в схему была добавлена клавиатура, чтобы пользователь мог вводить параметры. У клавиатуры 4 строки и 3 столбца. Пусть строки будут подключены к портам P2.0 – P2.3, а столбцы – к P2.4 – P2.6 соответственно.

На самом деле второй порт очень удобен для подключения клавиатуры, потому что он не требует внешних подтягивающих резисторов, и он проще в подключении. Если, например, использовать вышестоящий нулевой порт, то пришлось бы подключать сверху резисторы для стабильного уровня сигнала, потому что изначально этот порт используется как двунаправленный.

Далее в схему был добавлен ЦАП. Младшие разряды (D0-D3) подключены к земле, потому необходимо использовать только 8 разрядов из 12. Поэтому они и фиксируются в нуле, чтобы не влиять на результат преобразования. Также это сделано, чтобы они не висели незадействованными. Незадействованные входы могут ловить помехи, что может негативно сказаться на выходном сигнале.

К входам D4 – D11 подключён первый порт микроконтроллера. Подключение старших разрядов позволяет микроконтроллеру управлять более значимыми битами данных, что, в теории, должно повышать точность

генерируемого аналогового сигнала. Первый порт – стандартный восьмибитный двунаправленный порт с встроенными подтягивающими резисторами, что делает его удобным для передачи данных.

К первым трём младшим битам третьего порта были подключены три соответствующих световых индикатора под каждый режим. Третий порт микроконтроллера поддерживает работу с периферийными устройствами и часто используется в случае, когда работа идёт с выводом управляющих сигналов. В данном случае они нужны для включения и выключения светодиодов в определённые моменты времени.

Пин микроконтроллера P3.6/$\overline{\text{WR}}$ подключён сразу к нескольким входам ЦАП, а именно: $\overline{LDAC}$, $\overline{WR}$, $\overline{CSLSB}$ и $\overline{CSMSB}$. Данный пин служит ключевым управляющим сигналом для передачи данных и контроля ЦАП. Этот сигнал синхронизирует процесс записи данных в ЦАП. Управление через $\overline{WR}$ и синхронизация с помощью $\overline{LDAC}$ в случае изменения параметров обеспечивает правильную передачу данных в ЦАП и точность в создании выходного сигнала.

Помимо этого, сигнал $P3.7/\overline{RD}$ подключён к каналу D добавленного в схему осциллографа. Этот сигнал служит для синхронизации работы схемы и отображения изменений состояния в случае ошибок.

Перейдём теперь к подключениям, связанным с ЦАП. Помимо ранее рассмотренных подключений, следует также отметить подключение $\overline{CLR}$ к батарее в схеме для постоянного удержания в активном состоянии. Это заставляет ЦАП не сбрасывать свои значения и пребывать в неактивном состоянии.

Выход ЦАП VOUT подключён к первому каналу осциллографа (A). Именно на этот канал будет передаваться итоговый синусоидальный сигнал и именно на этом канале будут видны все изменения сигнала в случае изменений его параметров.

Вход ЦАП REFHI подключён к батарее. Это означает, что опорное напряжение для положительной части выходного сигнала задаётся

напряжением батареи (5В). Это в целом стандартная практика, когда источник питания используется в качестве опорного напряжения, чтобы обеспечить максимальное значение аналогового сигнала, которое будет генерировать ЦАП.

Сигналы REFLO и X1/X2 подключены к земле, так как в проекте используется однонаправленное питание и не требуется отрицательное опорное напряжение для генерации аналогового сигнала. Подключение REFLO к земле гарантирует, что нижняя граница выходного сигнала будет на уровне 0 В. Таким образом, диапазон выходных значений будет от 0 В до REFHI (5В). Также подключение выводов X1/X2 именно к земле связано с отсутствием необходимости во внешнем осцилляторе, то бишь схема работает на внутреннем тактовом генераторе.

К каждому из световых индикаторов подключены, как отмечалось ранее, резисторы с сопротивлением 0.1 Ом, а те, в свою очередь, в совокупности подключены к земле. Так как светодиоды – полупроводниковые устройства, которые ограничивают ток через себя, то они требуют дополнительной защиты, чтобы не повредиться из-за слишком большого тока. Без резистора светодиод может перегреться и выйти из строя. После этого идёт подключение к земле для замыкания цепи.

1.4. Значение элементов

Подытожим всё вышесказанное в контексте значимости в схеме. Микроконтроллер является центральным элементом управления в проекте. Он выполняет обработку данных и управление всеми подключенными устройствами, включая клавиатуру, ЦАП, светодиоды и осциллограф. Клавиатура используется для ввода данных, таких как амплитуда, частота и фаза синусоидального сигнала, то есть обеспечивает пользовательский интерфейс для настройки параметров сигнала. ЦАП преобразует цифровые данные, полученные от микроконтроллера, в аналоговый сигнал, который может быть использован для генерации синусоидального сигнала. Также он отвечает за точность и качество генерируемого аналогового сигнала.

Осциллограф используется для визуализации выходного сигнала на экране, чтобы проверить правильность генерируемой синусоидальной волны. Светодиоды используются для индикации текущего состояния устройства, то есть для отображения режима амплитуды, частоты или фазы. Резисторы ограничивают ток, протекающий через светодиоды, чтобы предотвратить их повреждение, а батарея обеспечивает питание всей схемы.

2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В этом разделе будет подробно рассмотрено программное обеспечение, которое управляет работой микроконтроллера и взаимодействует с аппаратной частью устройства. Программа обеспечивает управление всеми функциями генератора синусоидальных сигналов, включая настройку амплитуды, частоты и фазы сигнала через клавиатуру, а также управление выводом сигналов на ЦАП.

2.1. Назначение программы и её основные функции

Разработанная программа управляет генерацией синусоидальных сигналов с возможностью настройки амплитуды, частоты и фазы. Пользователь вводит значения для этих параметров через клавиатуру, а программа преобразует их в соответствующие цифровые значения и передает на ЦАП для генерации аналогового сигнала. Основными функциями программы являются:

1. Чтение ввода с клавиатуры для изменения параметров сигнала.
2. Генерация синусоидального сигнала с учётом введённых пользователем параметров.
3. Управление светодиодами для отображения состояния параметров.
4. Управление циклом работы генератора и управление выводом сигнала через ЦАП.

2.2. Процесс разработки программы

В качестве программы был выбран Keil Vision, так как в качестве языка программирования был также выбран C, который напрямую не поддерживается Proteus. Помимо этого, разработка производилась с использованием библиотеки REG51.h, которая содержит определения для работы с портами микроконтроллера серии 8051. Если говорить в общем, то процесс разработки включал:

1. Проектирование интерфейса для ввода данных с клавиатуры, обработки значений амплитуды, частоты и фазы.

2. Создание алгоритмов для генерации синусоидального сигнала с использованием математической функции синуса.

3. Настройка выводов микроконтроллера для управления светодиодами, отображающими текущий режим работы.

4. Реализация функции задержки для управления частотой генерации сигнала.

Тестирование программы производилось в Proteus, куда предварительно был загружен скомпилированный в Keil Vision hex-файл.

2.3. Обобщённая структура программы

В обобщённой структуре программы можно выделить несколько частей. Рассмотрим их подробнее.

1. Инициализация. На данном этапе происходит настройка портов микроконтроллера под различные задачи. Например, в программе выделяется и настраивается второй порт для работы с клавиатурой (согласно подключению в схеме), чтобы обрабатывать сигналы от строк и столбцов клавиш. Каждая клавиша на клавиатуре будет соответствовать определённому значению, которое программа сможет распознавать.

Также, первые три входа третьего порта, к которым в схеме подключены светодиоды, инициализируются для индикации режима. В дальнейшем они будут сигнализировать о том, какой из параметров в данный момент изменяется.

Помимо этого, определяются ещё и переменные для хранения текущих значений амплитуды, частоты, фазы, а также переменные для работы с временными задержками и отсчётами. Эти переменные необходимы в регулировании параметров сигнала.

2. Основной цикл программы. Программа содержит основной цикл, в котором выполняется непрерывное обновление и выполнение всех необходимых функций. Программа периодически проверяет нажатие клавиш, определяя, какой параметр пользователь хочет изменить. В зависимости от нажатой клавиши, программа вызывает соответствующую функцию для

обработки ввода. На каждом шаге цикла программа рассчитывает текущее значение синусоидального сигнала на основе установленных параметров (либо пользователем, либо заранее прописанными в начале программы для начального вывода). После расчёта значения синусоидального сигнала программа отправляет данные на ЦАП через первый порт для их преобразования в аналоговый сигнал.

3. Функции для ввода параметров. Для ввода и изменения параметров синусоидального сигнала используются следующие функции:

1. **inputAmplitude.** Эта функция обрабатывает ввод значений амплитуды с клавиатуры. Пользователь может ввести желаемое значение амплитуды, после чего оно сохраняется в соответствующей переменной. Функция также управляет светодиодом, указывающим на активный режим.

2. **inputFrequency.** Функция позволяет вводить значения частоты с клавиатуры, которые сохраняются в соответствующей переменной. В момент выбора этой функции включается соответствующий светодиод, показывающий, что изменяется частота.

3. **inputPhase.** Функция обрабатывает ввод значения фазы. Введённое значение сохраняется в соответствующей переменной, и при этом включается светодиод, сигнализирующий о режиме настройки фазы.

4. Генерация синусоидального сигнала. Синусоидальный сигнал генерируется в основном цикле программы. Сначала идёт расчёт значения сигнала. Используя текущие значения амплитуды, частоты и фазы, программа вычисляет значение синусоидального сигнала для каждого шага с использованием функции sin() из библиотеки math.h. Затем идёт формирование выходного сигнала. Рассчитанное значение сигнала отправляется на первый порт микроконтроллера, к которому подключен ЦАП. ЦАП преобразует цифровое значение в итоговый аналоговый синусоидальный сигнал.

5. Управление светодиодами. Светодиоды используются для индикации текущего режима настройки. Первый светодиод красного цвета

включается при изменении амплитуды и отключается, когда ввод завершён. Аналогично работают и два других светодиода: зелёный для частоты и голубой для фазы.

2.4. Алгоритмы работы программы

В программе можно выделить несколько конкретных алгоритмов работы. Рассмотрим каждый из них подробнее.

Первым алгоритмом является алгоритм обработки ввода с клавиатуры. Он выполняет следующие шаги для получения значений параметров сигнала. Сначала программа организует считывание данных с клавиатуры, подключенной ко второму порту, а именно по строкам и столбцам. Каждый цикл сканирования начинается с установки всех строк в высокий уровень (логическая единица) и последующего поочерёдного перевода каждой строки в низкий уровень (логический ноль). Затем идёт проверка состояния столбцов. Когда одна из строк устанавливается в низкий уровень, программа проверяет каждый из столбцов, чтобы определить, нажата ли какая-либо из клавиш в текущей строке. Если нажатие обнаружено, программа фиксирует, какая именно клавиша была нажата, и завершает проверку столбцов для данной строки.

Далее определяется обработка нажатой клавиши. После определения строки и столбца, программа использует матрицу символов для получения символа нажатой клавиши. Если это цифра, то программа интерпретирует её как часть значения амплитуды, частоты или фазы, добавляя к накопленному значению. Завершающим шагом является ожидание подтверждения выбора. Если пользователь нажимает соответствующую клавишу «#», программа воспринимает это как сигнал завершения ввода значения. В этом случае ввод прекращается, значение сохраняется в соответствующей переменной, и программа возвращается в основной цикл. Данный алгоритм позволяет удобно считывать многозначные значения и управлять изменением параметров в реальном времени.

Вторым алгоритмом является алгоритм генерации искомого синусоидального сигнала. Он выполняет вычисления, необходимые для создания синусоидального сигнала с заданными параметрами. Начинается всё с задания фиксированного шага. В программе используется константа, которая определяет количество дискретных шагов, из которых состоит один период синусоиды. Это позволяет избежать резких изменений в форме сигнала и способствует его плавной генерации. После этого идёт вычисление значения сигнала. В каждом шаге основной цикл программы вычисляет текущее значение синусоидального сигнала с использованием математической функции синуса из соответствующей библиотеки, подставляя текущие параметры. Формула имеет вид:

$$\text{out} = \frac{\text{amplitude}}{2} * \left(1 + \sin\left(\frac{2\pi \text{i} * \text{frequency}}{\text{FIXED}_{\text{STEPS}}} + \text{phase}\right)\right),$$

где i – текущий шаг, amplitude – амплитуда, frequency - частота, а phase – фазовый сдвиг.

Затем идёт отправка на ЦАП. Полученное значение передаётся на первый порт микроконтроллера, который подключён к цифро-аналоговому преобразователю (ЦАП). ЦАП преобразует это значение в аналоговый сигнал, который затем можно наблюдать на осциллографе.

Для регулирования частоты обновления значений синусоиды программа добавляет задержку с использованием соответствующей функции. Это позволяет контролировать скорость генерации синусоиды и гарантировать стабильное отображение сигнала на выходе. Этот алгоритм позволяет генерировать синусоидальный сигнал с заданными пользователем параметрами и отправлять его на ЦАП для последующего преобразования в аналоговый вид.

Третьим и последним алгоритмом является алгоритм управления светодиодами. Он используется для индикации текущего режима (изменение амплитуды, частоты или фазы), позволяя пользователю понимать, какой параметр настраивается. Начинается он с активации светодиодов. Затем идёт

логика включения и выключение светодиодов в соответствии с текущими условиями и временем, а именно: при вызове соответствующей функции (для каждого светодиода она своя), светодиод, связанный с этим параметром, включается, чтобы проинформировать пользователя об активном режиме изменения параметра. Например, при вызове функции inputeAmplitude включается первый, красный светодиод. При вызове inputFrequency – второй, зелёный, а при вызове inputPhase – третий, голубой. После завершения ввода параметра светодиод гаснет, показывая, что настройка данного параметра завершена. Это происходит при нажатии клавиши #, завершающей процесс ввода.

Этот алгоритм значительно облегчает работу пользователя, позволяя визуально отслеживать режимы настройки параметров и избегать ошибок при изменении значений.

3. ТЕСТИРОВАНИЕ И ОТЛАДКА АППАРАТНО-ПРОГРАММНОГО КОМПЛЕКСА

На данном этапе тестирования и отладки проводится проверка функциональности аппаратно-программного комплекса, что позволяет подтвердить соответствие устройства заданным техническим требованиям. Тестирование охватывает все основные функции проекта, включая корректность обработки сигналов, реагирование на команды пользователя и стабильность работы устройства. Для подтверждения правильности выполнения всех функций используются скриншоты и фотографии, иллюстрирующие результаты работы устройства на каждом этапе тестирования.

3.1. Описание проверки функций устройства

На данном этапе тестирования проверяются основные функции генератора, чтобы удостовериться, что они работают корректно и соответствуют техническим требованиям.

Нажимая на клавиши, пользователь вводит значения амплитуды, частоты и фазы. Программа должна корректно считывать и обрабатывать ввод, переключаться между режимами, а также сохранять и применять введенные значения к сигналу. Каждый ввод должен завершаться нажатием клавиши «#» для фиксации значений.

Введенные параметры должны корректно отражаться на выходном сигнале, что должно подтверждаться через осциллограф. Например, увеличение амплитуды должно приводить к пропорциональному увеличению амплитуды сигнала на осциллографе, изменение частоты – к изменению частоты синусоидального сигнала, а изменение фазы – к изменению начальной точки вывода сигнала.

Для каждого изменяемого параметра установлены светодиоды, которые должны загораться при активации соответствующего режима. Это позволяет пользователю получать визуальную информацию о текущем состоянии устройства. Каждый из светодиодов должен загораться в ответ на ввод

значения параметра и гаснуть при его завершении, показывая завершение работы с параметром.

3.2. Скриншоты

Проверим работоспособность устройства. Запустим разработанную схему с предварительно загруженным hex-файлом с исходным кодом. На первом канале осциллографа можно увидеть вывод синусоиды с начально заданными параметрами. Это показано на рисунке 12.

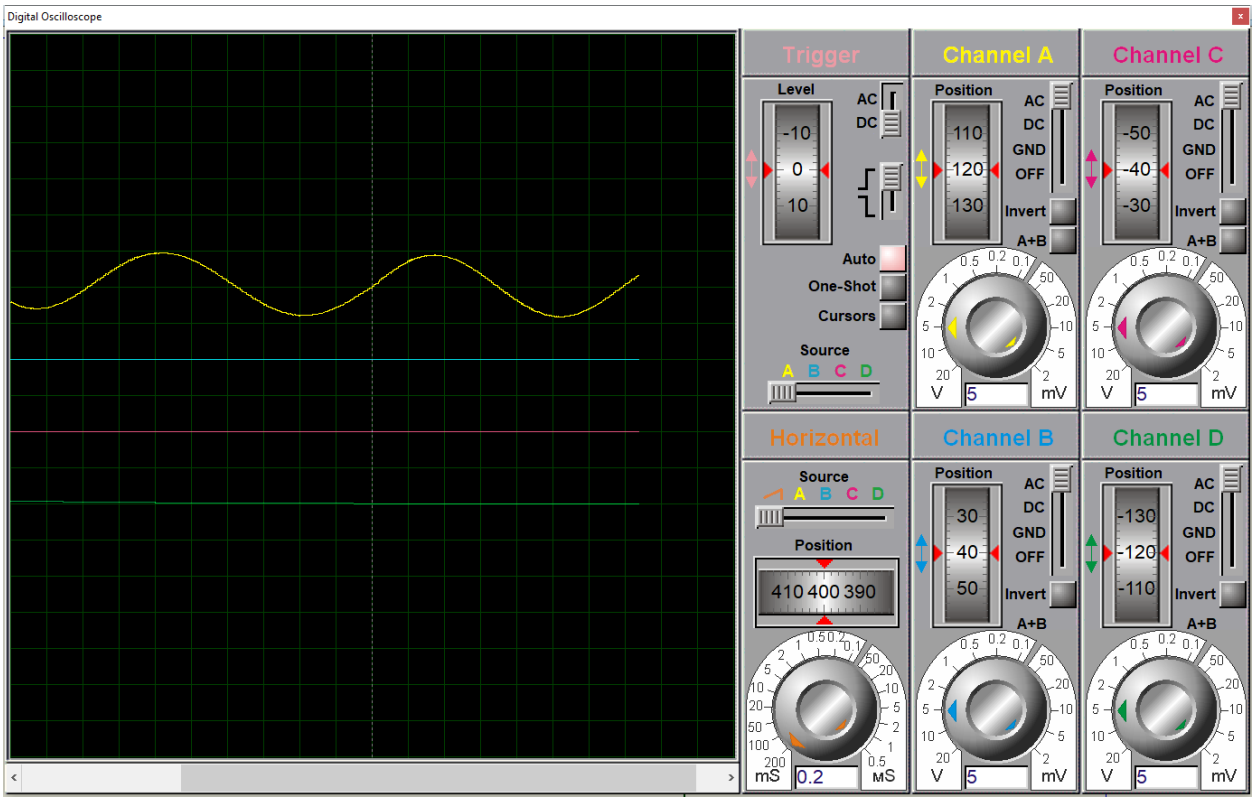

Рисунок 12 – Начальный вывод

Теперь пользователь может в реальном времени менять характеристики синусоиды. Например, зададим новое значение амплитуды. Отметим, что изначально значение было равно 200. Уменьшим его на 100. Сначала нажав соответствующую клавишу «*» для перехода в режим редактирования (при первом нажатии это будет режим редактирования амплитуды). На осциллографе можно заметить вывод постоянного нуля – визуальное подтверждение того, что идёт режим ввода нового параметра. Это показано на рисунке 13.

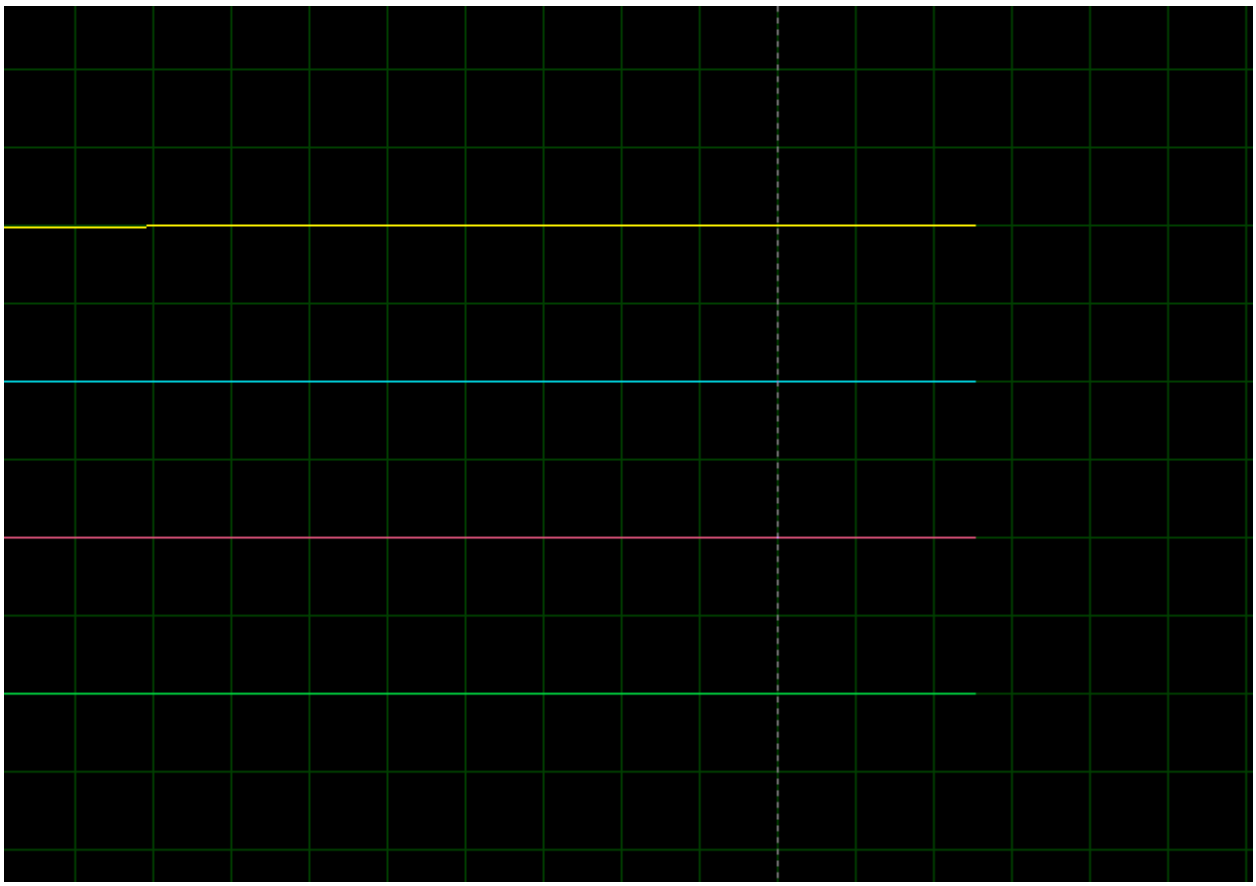

Рисунок 13 – Режим редактирования

При этом на схеме можно заметить, что загорелась первая лампочка (красная). Это показано на рисунке 14.

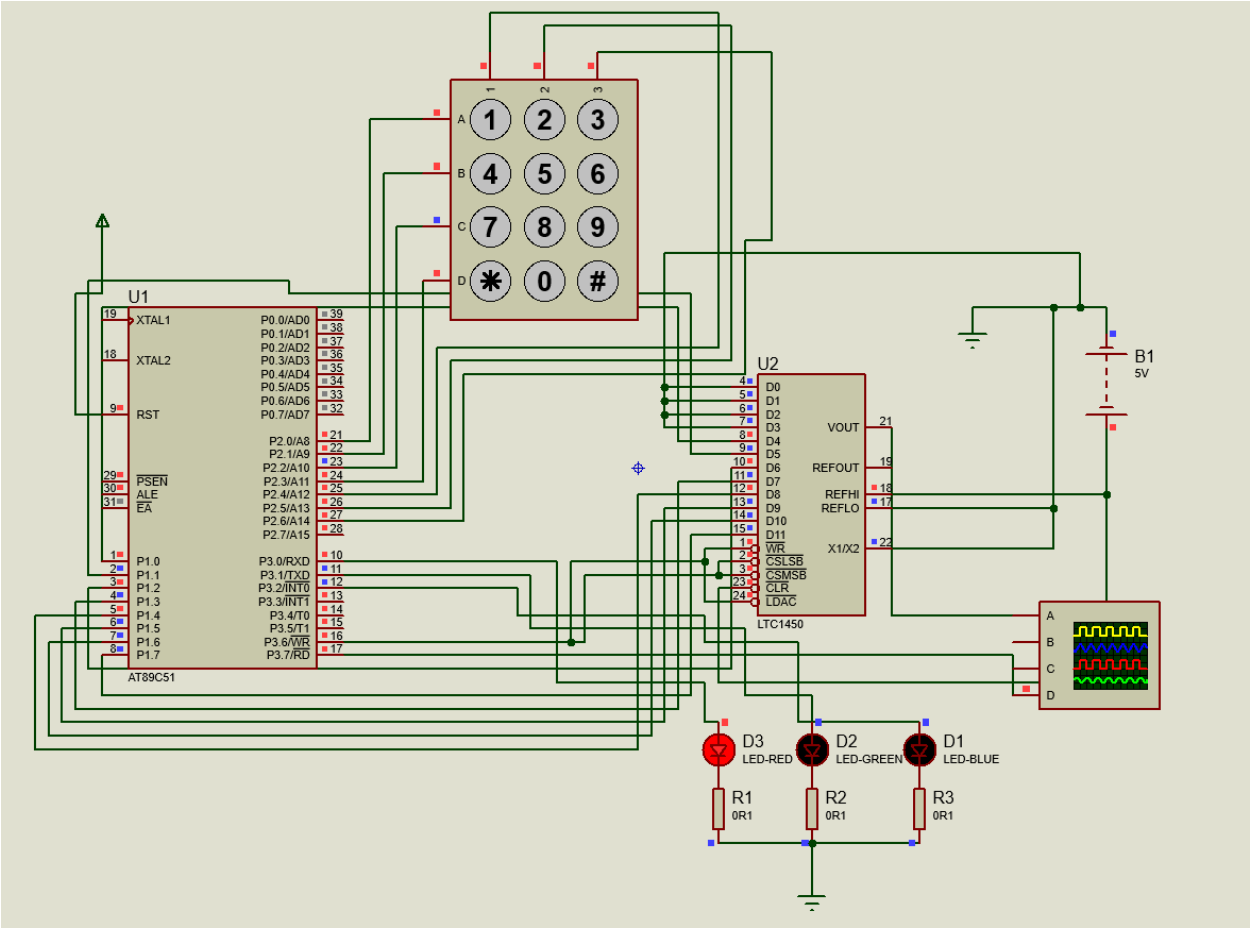

Рисунок 14 – Режим амплитуды

Это означает, что если пользователь введёт какое-нибудь значение, то поменяется именно амплитуда. Если же он захочет изменить другой параметр в случае первого изменения, то ему необходимо ничего не вводить, а просто нажать на решётку, чтобы перейти на другой режим. После ввода можно увидеть результат на осциллографе, как это показано на рисунке 15.

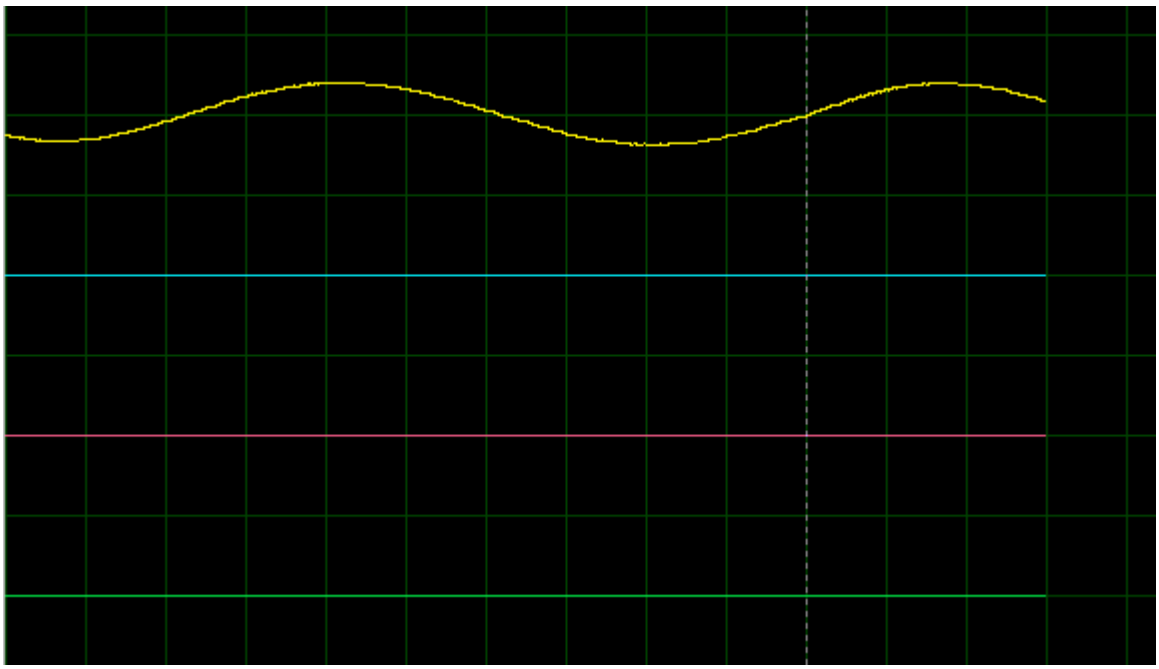

Рисунок 15 – Синусоида с новой амплитудой

Пусть, например, далее пользователь хочет поменять ещё и частоту. Изначально она была равна 2. Увеличим её в 5 раз. После повторного нажатия на звёздочку зажжётся уже следующая лампочка – зелёная. При этом снова вывод на осциллографе будет выдавать постоянную, так как снова осуществиться переход в режим редактирования. Это показано на рисунке 16.

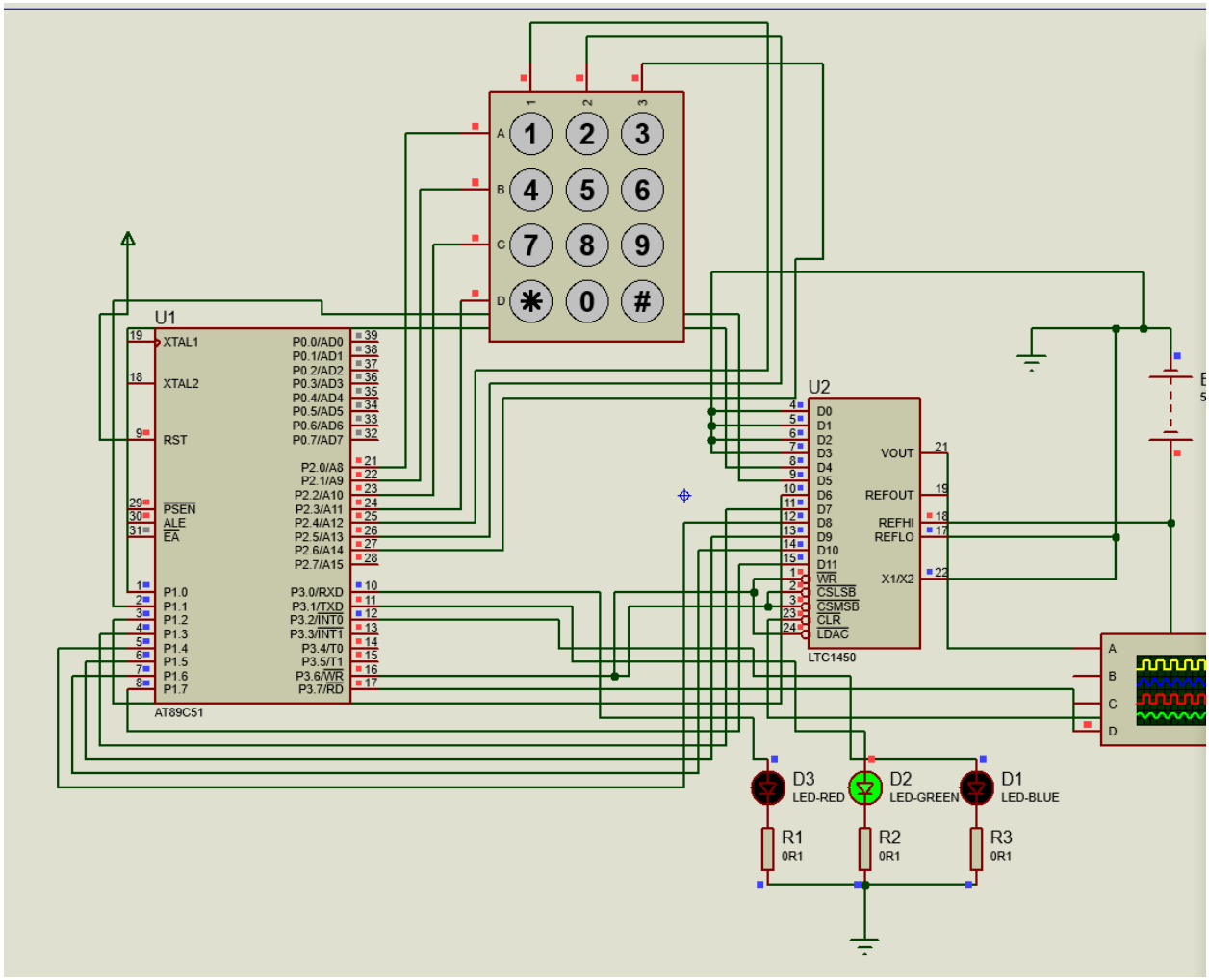

Рисунок 16 – Режим частоты

После соответствующего ввода на клавиатуре и последующего нажатия на решётку синусоида станет выводиться чаще, как это показано на рисунке 17.

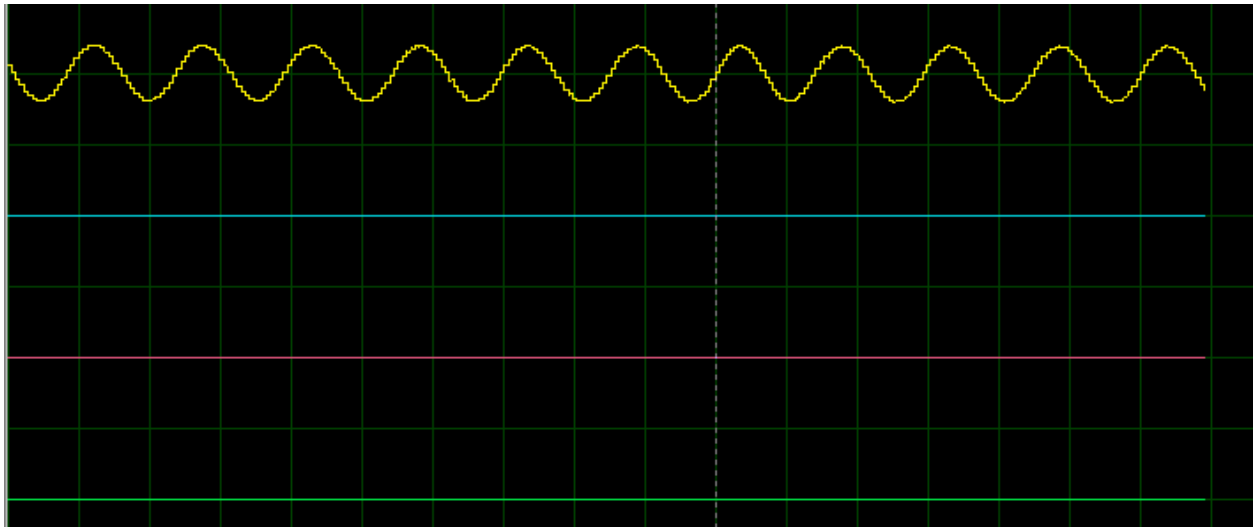

Рисунок 17 – Синусоида с новой частотой

Наконец, если пользователь захотел задать новое значение фазы, то необходимо снова нажать на звёздочку. В таком случае зажжётся голубая лампочка, как это показано на рисунке 18. На осциллографе снова будет вывод постоянной. Пусть новая фаза будет 270 градусов (в программе учтён перевод в радианы, а пользователь должен вводить именно градусное значение). Тогда синусоида будет начинаться из другой точки, как это показано на рисунке 19.

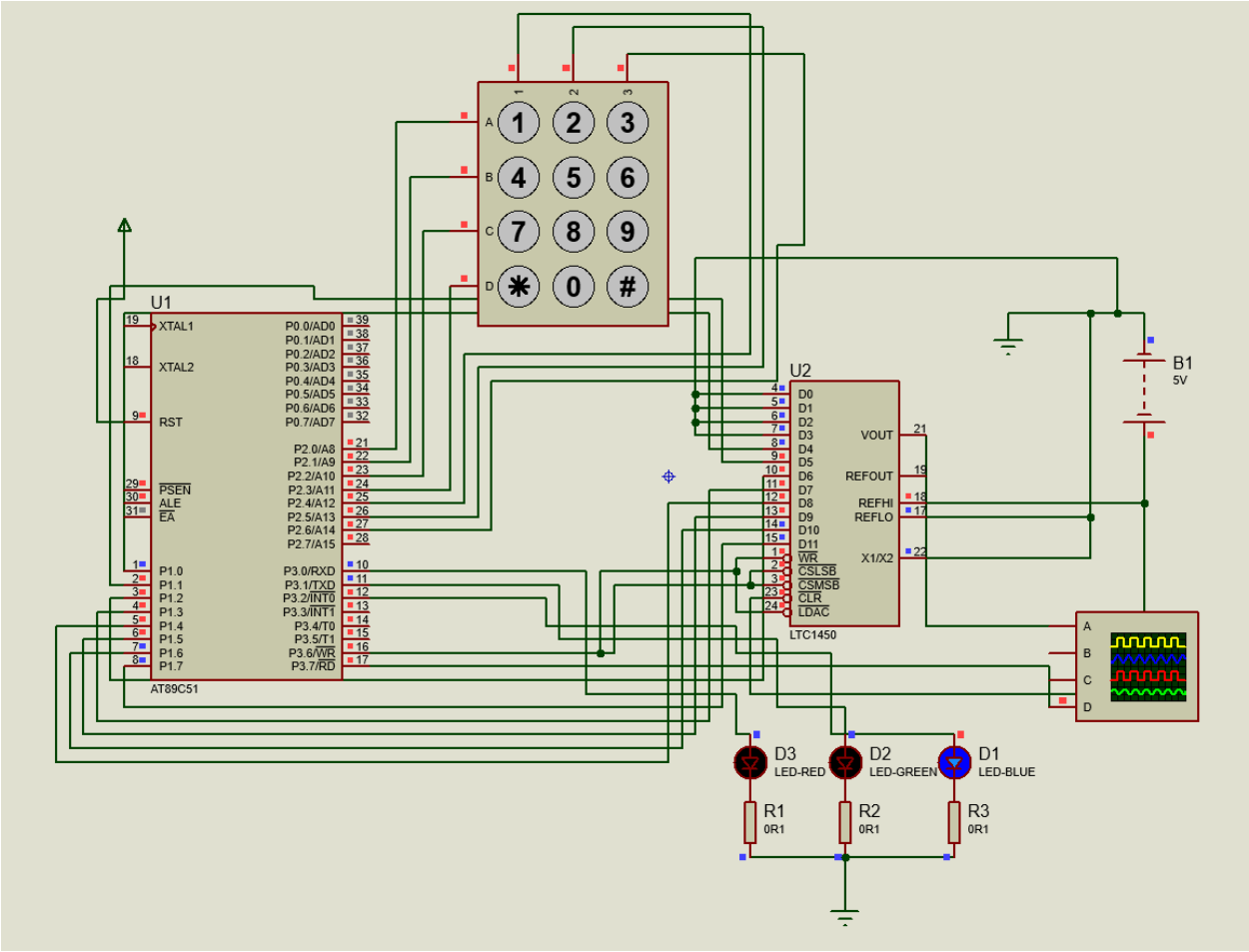

Рисунок 18 – Режим фазы

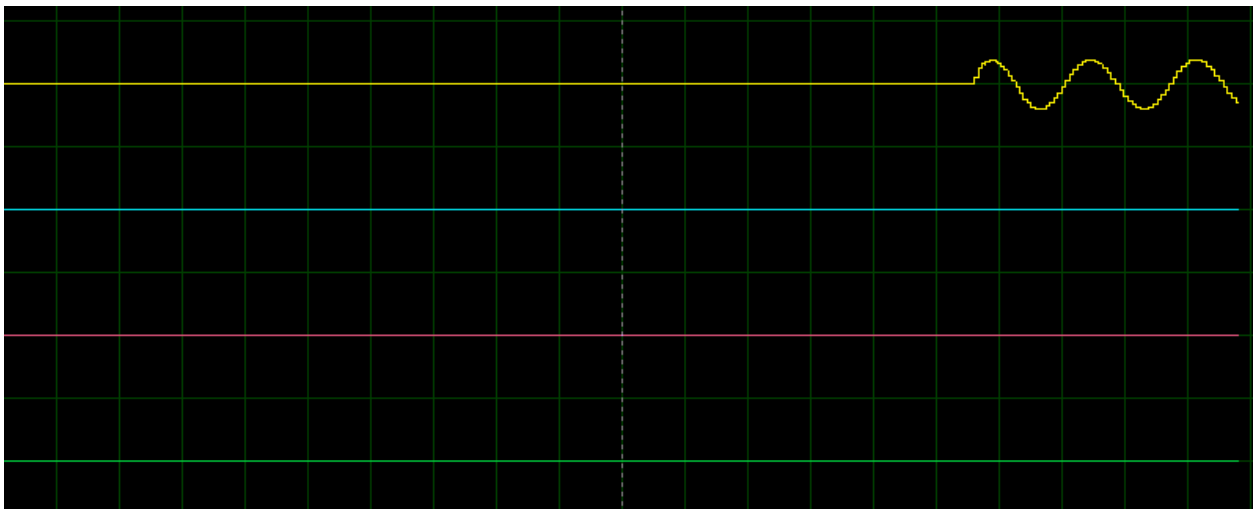

Рисунок 19 – Синусоида с новой фазой

Даже если пользователь передумает и захочет снова поменять какой-нибудь параметр, то он может снова нажать на звёздочку. «Проходиться» так по всем режимам можно до бесконечности.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано устройство, а именно - генератор синусоидальных сигналов, работающий на микроконтроллере серии 8051, с возможностью регулировки амплитуды, частоты и фазы выходного сигнала. Основное назначение устройства — генерировать аналоговый синусоидальный сигнал с заданными параметрами для целей тестирования и калибровки различных электронных устройств, а также, например, для использования в учебных и лабораторных экспериментах.

Устройство выполняет ряд важных функций, необходимых для гибкого и точного управления синусоидальным сигналом. Оно позволяет пользователю посредством встроенной клавиатуры задавать параметры сигнала — амплитуду, частоту и фазу, что делает устройство удобным и универсальным. Управление этими параметрами отображается на трёх светодиодных индикаторах, которые сигнализируют о текущем режиме настройки, упрощая пользователю работу с устройством и делая его более наглядным. Параметры сигнала обрабатываются программой микроконтроллера, которая передает значения на цифро-аналоговый преобразователь (ЦАП) и выводит сигнал на осциллограф для визуализации и дальнейшего анализа.

Основные технические характеристики разработанного устройства включают:

1. *Диапазон амплитуды*. Амплитуда сигнала регулируется в пределах до 2,5В, так как ЦАП имеет диапазон входных напряжений, ограниченный напряжением питания, которое на схеме составляет 5В, отчего с учётом колебаний вокруг нулевого уровня для синусоиды и получается максимальное значение, равное 2,5В. Однако значения можно интерпретировать и через обычное числовое значение, которое будет равно 0 – 255 для 8-битной шкалы, потому что именно такой формат используются на ЦАП.
2. *Диапазон частоты*. Данный диапазон зависит от частоты тактового сигнала микроконтроллера и алгоритма генерации синусоиды. Следует

отметить, что каждый цикл задержки во внешнем цикле выполняет 120 инкрементов внутреннего цикла, тогда как внешний цикл выполняется мс раз (количество миллисекунд задержки). Значит, общее количество тактов, необходимых для одной задержки в миллисекундах равно:

$$\text{Количество тактов} = 120 * \text{мс}$$

Используемый микроконтроллер работает на тактовой частоте 12 МГц. Отсюда можем найти время каждого такта:

$$\text{Время одного такта} = \frac{1}{12\ \text{МГц}} = \frac{1}{12\ 000\ 000} = 83.33\ \text{нс}$$

Теперь, чтобы вычислить реальное время задержки, необходимо умножить количество тактов на время одного такта:

$$\text{Время задержки (с)} = \frac{\text{Количество тактов}}{\text{Частота}} = \frac{120 * \text{мс}}{12\ 000\ 000} =$$

$$= 10^{-5} * \text{мс секунд}$$

Время одного цикла синусоиды зависит от того, сколько времени занимает выполнение полного цикла в 255 шагах. Каждый шаг в цикле синусоиды будет задерживаться на delayTime миллисекунд. То есть, для одного периода (255 шагов) задержка составит:

$$\text{Частота} = \frac{1}{255 * delayTime * 10^{-3}(\text{из мс в с})} = \frac{1}{255 * 5 * 10^{-3}} = 0.784\ \text{Гц}$$

Грубо говоря, диапазон частот зависит от диапазона значений переменной *delayTime*. Если он будет от 1 до 10, то диапазон частот будет соответственно:

$$\text{Частота}\ (maximum) = \frac{1}{255 * 1 * 10^{-3}} = 3.92\ \text{Гц}$$

$$\text{Частота}(minimum) = \frac{1}{255 * 10 * 10^{-3}} = 0.392\ \text{Гц}$$

Конечно же, следует понимать, что чем больше значение задержки, тем меньше значение частоты. Конечно, можно подогнать диапазон значений *delayTime* и под конкретный формат сигнала, например, аудио, который

должен находиться в диапазоне от 0.1 Гц до 10 кГц. Тогда диапазон значений для этой переменной будет от 0.392 мс до 39.2 мс.

3. *Диапазон фазы*. Так как пользователь вводит значения в градусах, то ограничение составляют значения 0 – 360.

4. *Пользовательский интерфейс.* Устройство оснащено цифровой клавиатурой для ввода параметров и светодиодными индикаторами для отображения текущего режима работы.

5. *ЦАП.* Используется для преобразования цифрового сигнала, формируемого микроконтроллером, в аналоговый синусоидальный сигнал, подаваемый на осциллограф.

Область применения разработанного устройства достаточно широка. Оно может использоваться в лабораторной практике при обучении основам цифровой электроники, микроконтроллерных систем и сигналов, в качестве тестового генератора для проверки работы электронных схем, а также в условиях сервисного центра для калибровки и тестирования оборудования. Благодаря возможности изменения параметров сигнала, данный генератор также может применяться в научных экспериментах, требующих использования точного синусоидального сигнала с регулируемыми характеристиками.

Таким образом, разработанный генератор синусоидальных сигналов обладает широкими функциональными возможностями и может найти применение в самых различных областях, включая учебные, исследовательские и сервисные задачи.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ATMEL. AT89C51: 8-битный микроконтроллер с 4 КБ программируемой флеш-памятью. Официальная документация. [Электронный ресурс]. – URL: https://electronix.org.ru/datasheet/ATMEL/AT89C51.PDF?ysclid=m3m213e3i2800175185 (дата обращения: 17.11.2024).

2. Linear Technology. LTC1450: 12-битный ЦАП с интерфейсом SPI. Официальная документация. [Электронный ресурс]. – URL: LTC1450/LTC1450L - Parallel Input, 12-Bit Rail-to-Rail Micropower DACs in SSOP (дата обращения: 17.11.2024).

3. Keil. Getting Started with Keil C51: Programming and Debugging for 8051 Microcontrollers. Руководство пользователя. [Электронный ресурс]. – URL: https://www.keil.com/dd/docs/c51/ (дата обращения: 17.11.2024).

4. Sarma, G. R. L. 8051 Microcontroller: Programming and Interfacing. Официальное руководство. [Электронный ресурс]. – URL: https://www.electronicwings.com/8051 (дата обращения: 17.11.2024).

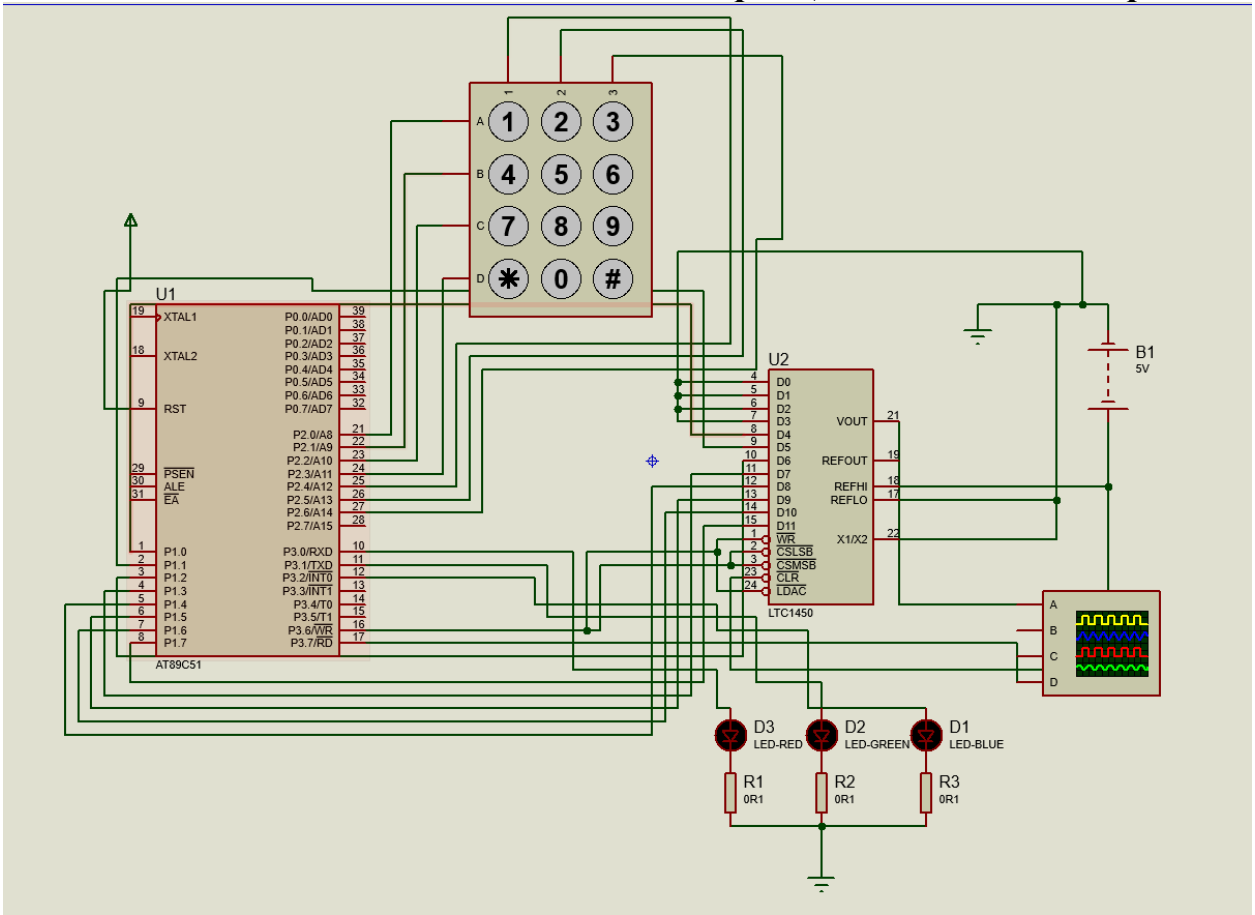

Рисунок А.1 – Принципиальная схема проекта

```
#include <REG51.h>  // Подключаем заголовочный файл для работы с микроконтроллером 8051
#include <stdio.h>   // Подключаем библиотеку для стандартного ввода-вывода (для функций, таких как printf)
#include <math.h>    // Подключаем библиотеку математических функций (для функции sin)

// Определяем макросы для фиксированных значений
#define FIXED_STEPS 255  // Количество шагов для одного цикла синусоиды
#define PI 3.14159265     // Константа Пи для использования в вычислениях синуса

// Объявляем переменные
unsigned char count;     // Счётчик для перехода между режимами
xdata unsigned char out; // Выходное значение для ЦАП
unsigned char amplitude = 220; // Значение амплитуды сигнала
unsigned char frequency = 2;   // Значение частоты сигнала
unsigned char phase = 0;       // Значение фазы сигнала
unsigned int delayTime = 5;    // Задержка для каждого шага синусоиды (в миллисекундах)

// Массив для представления клавиш на клавиатуре
char keys[4][3] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};

// Определяем выводы для строк и столбцов клавиатуры
sbit r1 = P2 ^ 0;
sbit r2 = P2 ^ 1;
sbit r3 = P2 ^ 2;
sbit r4 = P2 ^ 3;

sbit c1 = P2 ^ 4;
sbit c2 = P2 ^ 5;
sbit c3 = P2 ^ 6;

// Определяем выводы для светодиодов
sbit ledAmplitude = P3 ^ 0;
sbit ledFrequency = P3 ^ 1;
sbit ledPhase = P3 ^ 2;

// Функция задержки, принимающая миллисекунды
void delay(unsigned int ms) {
    unsigned int i, j; // Переменные для обозначения внутреннего и внешнего циклов
    for (i = 0; i < ms; i++) // Внешний цикл для миллисекунд
        for (j = 0; j < 120; j++); // Внутренний цикл для тактовых импульсов, создающий задержку
}

// Функция для получения нажатой клавиши с клавиатуры
char getKey() {
    unsigned char row; // Переменная, хранящая значение строки

    for (row = 0; row < 4; row++) { // Проходим по всем строкам клавиатуры
        r1 = r2 = r3 = r4 = 1; // Подтягиваем все строки в 1 (высокий уровень), для предотвращения конфликта между собой при считывании ввода
        switch (row) {           // Выбираем одну строку для сканирования
        case 0: r1 = 0; break;
        case 1: r2 = 0; break;
        case 2: r3 = 0; break;
        case 3: r4 = 0; break;
        }
```

```
        // Проверяем, какая клавиша была нажата в текущей строке
        if (c1 == 0) { while (c1 == 0); return keys[row][0]; }
        if (c2 == 0) { while (c2 == 0); return keys[row][1]; }
        if (c3 == 0) { while (c3 == 0); return keys[row][2]; }
    }

    return 0; // Если клавиша не была нажата, возвращаем 0
}

// Функция для ввода значения амплитуды с клавиатуры
void inputAmplitude() {
    unsigned char inputValue = 0;  // Переменная для хранения введённого значения
    char key; // Переменная для хранения символа, полученного с клавиатуры
    int x = 0;  // Флаг для отслеживания, был ли введён хотя бы один символ

    while (1) {  // Бесконечный цикл, пока не введено корректное значение
        key = getKey();  // Получаем нажатую клавишу
        if (key != 0) {  // Если клавиша нажата
            if (key >= '0' && key <= '9') {  // Если нажата цифра
                inputValue = (inputValue * 10) + (key - '0');  // Обновляем введённое
значение
                x = 1; // Устанавливаем флаг
            }
            else if (key == '#' && x == 1) {  // Если нажата решётка после ввода числа
                ledAmplitude = 0;  // Выключаем светодиод амплитуды
                count = 0;  // Сбрасываем счётчик
                amplitude = inputValue;  // Записываем введённое значение амплитуды
                break;  // Выходим из цикла
            }
            else if (key == '#' && x == 0) {  // Если нажата решётка до ввода числа
                ledAmplitude = 0;  // Выключаем светодиод
                break;  // Выходим из цикла
            }
        }
    }
}

// Функция для ввода значения частоты с клавиатуры
void inputFrequency() {
    unsigned int inputValue = 0;  // Переменная для хранения введённого значения
    char key; // Переменная для хранения символа, полученного с клавиатуры
    int x = 0;  // Флаг для отслеживания, был ли введён хотя бы один символ

    while (1) {  // Бесконечный цикл, пока не введено корректное значение
        key = getKey();  // Получаем нажатую клавишу
        if (key >= '0' && key <= '9') {  // Если нажата цифра
            inputValue = (inputValue * 10) + (key - '0');  // Обновляем введённое
значение
            x = 1; // Устанавливаем флаг
        }
        else if (key == '#' && x == 1) {  // Если нажата решётка после ввода числа
            ledFrequency = 0;  // Выключаем светодиод частоты
            count = 0;  // Сбрасываем счётчик
            frequency = inputValue;  // Записываем введённую частоту
            inputValue = 0;  // Сбрасываем временную переменную
            break;  // Выходим из цикла
        }
        else if (key == '#' && x == 0) {  // Если нажата решётка до ввода числа
            ledFrequency = 0;  // Выключаем светодиод
            break;  // Выходим из цикла
        }
    }
}
```

```
// Функция для ввода значения фазы с клавиатуры
void inputPhase() {
    unsigned int inputValue = 0;  // Переменная для хранения введённого значения
    char key; // Переменная для хранения символа, полученного с клавиатуры
    int x = 0;  // Флаг для отслеживания, был ли введён хотя бы один символ

    while (1) {  // Бесконечный цикл, пока не введено корректное значение
        key = getKey();  // Получаем нажатую клавишу
        if (key >= '0' && key <= '9') {  // Если нажата цифра
            inputValue = (inputValue * 10) + (key - '0');  // Обновляем введённое
значение
            x = 1; // Устанавливаем флаг
        }
        else if (key == '#' && x == 1) {  // Если нажата решётка после ввода числа
            ledPhase = 0;  // Выключаем светодиод фазы
            count = 0;  // Сбрасываем счётчик
            phase = inputValue;  // Записываем введённую фазу
            inputValue = 0;  // Сбрасываем временную переменную
            break;  // Выходим из цикла
        }
        else if (key == '#' && x == 0) {  // Если нажата решётка до ввода числа
            ledPhase = 0;  // Выключаем светодиод
            break;  // Выходим из цикла
        }
    }
}

// Главная функция программы
void main(void) {
    unsigned int i;  // Переменная для циклов
    char key;  // Переменная для хранения нажатой клавиши
    int count = 0;  // Счётчик для переключения между режимами

    ledAmplitude = 0;  // Изначально светодиод амплитуды выключен
    ledFrequency = 0;  // Изначально светодиод частоты выключен
    ledPhase = 0;      // Изначально светодиод фазы выключен

    while (1) {  // Бесконечный цикл работы программы
        for (i = 0; i < FIXED_STEPS; i++) {  // Проходим по всем шагам синусоиды
            // Генерируем значение синусоидального сигнала с учётом амплитуды, частоты и
фазы
            out = (unsigned char)((amplitude / 2) * (1 + sin(2 * PI * i * frequency /
FIXED_STEPS + (phase * PI / 180))));
            P1 = out;  // Выводим значение на порт P1 (ЦАП)
            delay(delayTime);  // Ждём указанное время

            key = getKey();  // Проверяем, была ли нажата клавиша
            if (key == keys[3][0]) {  // Если нажата клавиша "*"
                count++;  // Увеличиваем счётчик
                if (count == 1) {  // Если первый раз нажата клавиша
                    ledAmplitude = 1;  // Включаем светодиод амплитуды
                    inputAmplitude();  // Вводим амплитуду
                }
                else if (count == 2) {  // Если второй раз нажата клавиша
                    ledFrequency = 1;  // Включаем светодиод частоты
                    inputFrequency();  // Вводим частоту
                }
                else if (count == 3) {  // Если третий раз нажата клавиша
                    count = 0;  // Сбрасываем счётчик
                    ledPhase = 1;  // Включаем светодиод фазы
                    inputPhase();  // Вводим фазу
                }
            }
```

```
            }
        }
    }
```