**Департамент образования мэрии города Новосибирска**

**Дворец творчества детей и учащейся молодёжи «Юниор»**

**Городской конкурс проектов учащихся 5-8-х классов**

Направление: научно-техническое

Секции: физика, информатика

**«****Оптимизация номиналов резисторов в схемах регулируемых источников питания»**

Автор проекта:

**Ларченко Владислав**

МАОУ «Лицей №159», 6 класс

Консультанты проекта:

**Еремина Виктория Игоревна**,

учитель математики

Контактный телефон

руководителя:

+7 923 155 56 59

г. Новосибирск, 2025

Оглавление

**Портфолио проекта4**

**Введение6**

Операционный усилитель7

Структура операционного усилителя8

Сопротивления при разных подключениях9

Последовательное подключение9

Параллельное подключение9

Задачи по теме исследования10

Реализация программы11

**Заключение12**

**Список использованной литературы13**

**Приложения14**

Портфолио проекта

1. **Название проекта:** «Оптимизация номиналов резисторов в схемах регулируемых источников питания»
2. **Фамилия, имя, отчество разработчика проекта:** Ларченко Владислав Алексеевич
3. **Класс:** 6 «М»
4. **Название, номер учебного учреждения, где выполняется проект:** МАОУ «Лицей № 159»
5. **Предметная область:** физика, информатика
6. **Время разработки проекта:** декабрь 2024 – март 2025
7. **Проблема:** При создании электроники, выпускающейся малыми партиями, часто требуется переконфигурация уровней напряжения источников питания. Такая переконфигурация выполняется путём замены номиналов резисторов. Требуется выбрать резисторы из имеющихся в наличии и при возможности снизить количество различных номиналов.
8. **Актуальность исследования:** Это исследование актуально при тестировании светочувствительных матриц (ПЗС), когда управляющие и питающие напряжения могут отличаться от партии к партии в процессе улучшения рабочих характеристик данных матриц.
9. **Цель работы:** Создать программу на языке программирования Python, которая будет предоставлять пользовательский интерфейс, позволяющий загружать список имеющихся в распоряжении резисторов и значения напряжений, которые будет выдавать источник питания, и рассчитывать номиналы сопротивлений в источниках питания.
10. **Объект исследования:** Язык программирования Python, принципиальная схема делителей напряжений, последовательные и параллельные схемы включения сопротивлений.
11. **Предмет исследования:** способ оптимизации значений по имеющемуся набору входных данных.
12. **Задачи работы:**
    1. Изучить научную литературу по данной проблеме;
    2. Найти и вывести формулы для расчёта сопротивлений резисторов;
    3. Изучить документацию к языку Python для поиска средств написания программы;
    4. Разработать алгоритм подбора по формулам для языка программирования Python.
    5. создать интерфейс взаимодействия программы с пользователем;
    6. добавить в код программы защиту от ошибок, неверного ввода чисел пользователем.
13. **Тип работы:** поисковый, исследовательский.
14. **Используемые технологии:** язык программирования Python (в среде программирования VScode)
15. **Форма продукта проекта:** презентация, научно-исследовательская работа, программа для поиска номиналов резисторов в схемах регулируемых источников питания.
16. **Исследование проекта:** Компьютерная реализация алгоритма для поиска номиналов резисторов на языке программирования Python.
17. **Область применения результата проекта:** физика, электроника.

Введение

***Актуальность исследования:*** Это исследование актуально при тестировании светочувствительных матриц (ПЗС), когда управляющие и питающие напряжения могут отличаться от партии к партии в процессе улучшения рабочих характеристик данных матриц.

***Проблема исследования:***При создании электроники, выпускающейся малыми партиями, часто требуется переконфигурация уровней напряжений источников питания. Такая переконфигурация выполняется путём замены номиналов резисторов. Требуется выбрать резисторы из имеющихся в наличии и при возможности снизить количество различных номиналов. Как можно перевести эти алгоритмы подбора на компьютерный язык? Как создать программу, которая будет решать подобные задачи значительно быстрее человека и без ошибок?

***Гипотеза исследования:*** Возможно ускорить подбор резисторов для схем регулируемых источников питания с помощью вычислительной техники.

***Цель исследования:***Создание компьютерной программы на языке программирования Python 3.9, которую можно будет использовать для решения задач по поиску резисторов для регулируемых источников питания.

***Объект исследования:*** язык программирования Python.

***Предмет исследования:*** выбранные для закодирования в программе формулы с помощью Python 3.9

***Задачи:***

1. Изучить научную литературу по данной проблеме;
2. Найти и вывести формулы для расчёта сопротивлений резисторов;
3. Изучить документацию к языку Python для поиска средств написания программы;
4. Перевести алгоритм подбора по формулам на язык программирования Python;
5. Создать интерфейс взаимодействия программы с пользователем;
6. Добавить в код программы защиту от ошибок, неверного ввода чисел пользователем.

Операционный усилитель

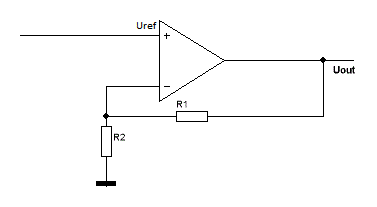
В качестве регулируемого источника питания в этом проекте я использовал операционный усилитель (ОУ), так как для изменения выходного напряжения достаточно лишь изменить сопротивление между выходом ОУ и инвертирующем входе, что и будет делать моя программа.

Операционный усилитель имеет два входа, инвертирующий (-) и неинвертирующий (+), и один выход. ОУ усиливает разность напряжений на входах.

Принцип действия ОУ наиболее понятно демонстрируется на модели «идеального операционного усилителя». Модель обладает следующими свойствами:  
Входы идеального ОУ не оказывают влияния на входные сигналы и имеют бесконечно большое сопротивление и бесконечно малую ёмкость.

1. Выход идеального ОУ имеет нулевое сопротивление и может обеспечить на нагрузке любое напряжение и любой ток.
2. Коэффициент передачи идеального ОУ стремится к бесконечности и не зависит от частоты входных сигналов.
3. Время задержки распространения сигнала в идеальном ОУ равно нулю, сдвиг фаз отсутствует.
4. Охваченный ООС идеальный ОУ стремится установить равное напряжение на входах.

Согласно первому свойству модели идеального операционного усилителя, входы ОУ ток не потребляют. А так как при такой схеме включения на инвертирующем входе ОУ стремится выровняться с напряжением на неинвертирующем входе (+), то падение напряжения на резисторе равно напряжению , а падение напряжения на последовательно включенных резисторах делителя напряжения и равно напряжению .



Структура операционного усилителя

Операционный усилитель — это многоступенчатая схема, состоящая из транзисторов, резисторов и иногда диодов.

**1. Дифференциальный входной каскад**

* **Цель:** Считывание и усиление разности напряжений между входами (инвертирующим и неинвертирующим).
* **Структура:**
  + Два транзистора (биполярных или полевых) с общим резистором или источником тока в эмиттерной/источной цепи.
  + Токовое зеркало (пара транзисторов) преобразует разностный ток в напряжение.
* **Эффект:** Высокий входной импеданс, подавление синфазных помех.

**2. Промежуточный каскад**

* **Цель:** Усиление сигнала.
* **Структура:**
  + Транзисторы в каскадной схеме (для увеличения усиления и полосы пропускания).
  + Резисторы задают рабочую точку, конденсатор — частотную коррекцию.

**3. Выходной каскад**

* **Цель:** Согласование с нагрузкой (низкий выходной импеданс).
* **Структура:**
  + Комплементарные транзисторы.
  + Диоды или транзисторы в диодном включении создают смещение для устранения перекрестных искажений.

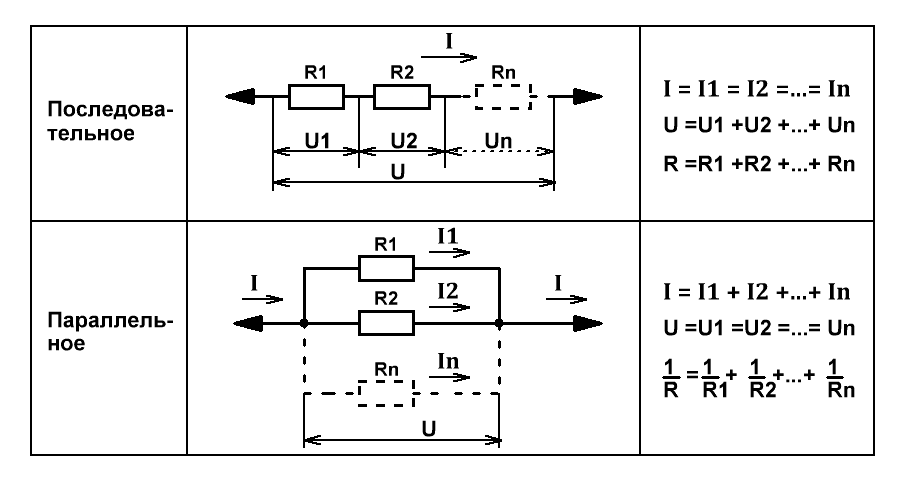
**Вспомогательные элементы:**

* **Цепи смещения:** Источники тока и резисторы для стабилизации режимов транзисторов.
* **Защита:** Диоды от переполюсовки питания.

ОУ объединяет высокое усиление (дифференциальный и промежуточный каскады) с мощным выходом, сохраняя стабильность за счет частотной коррекции и защиты. Внешние резисторы и конденсаторы задают его режим работы в реальных схемах.

Сопротивления при разных подключениях

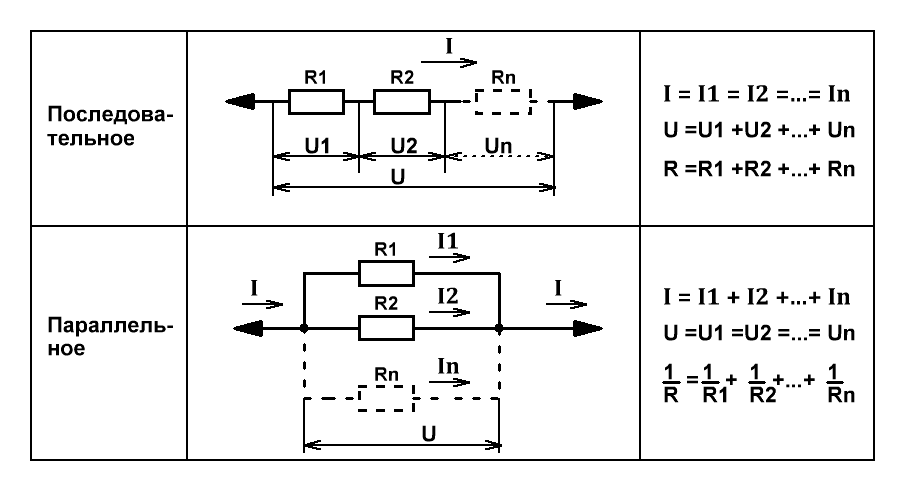
Последовательное подключение:

При последовательном соединении резисторов общее сопротивление равно сумме сопротивлений:

Параллельное подключение:

Напряжение одинаково: В параллельной цепи у всех резисторов вход в одних точках, поэтому на них падает равное напряжение *(U)*.

Ток через каждый резистор разный, определяется по формуле: , чем меньше сопротивление *(R)*, тем больше ток *(I)* через резистор, и наоборот.

Токи через резисторы обратно пропорциональны их сопротивлениям:

Задачи по теме исследования

***Задача 1:*** На неинвертирующий вход подаётся опорное напряжение (Vref) 2.5 В. Необходимо найти резисторы R1 и R2, чтобы на выходе получить (Vout) 8 В.

*Решение:*подбираем один резистор из общего списка (см. приложения), по формуле вычисляем второй и проверяем есть ли он в списке, если да, то эта пара резисторов – ответ.

При :

При :

*Ответ:*R1 = 1, R2 = 2.2; R1 = 10, R2 = 22

***Задача 2:*** На неинвертирующий вход подаётся опорное напряжение (Vref) 2.5 В. Необходимо найти резисторы R1 и R2, чтобы на выходе получить (Vout) 10.5 В.

*Решение:*

При :

Ответ:R1 = 7.5, R2 = 24.

***Задача 3:*** На неинвертирующий вход подаётся опорное напряжение (Vref) 2.5 В. Необходимо найти резисторы R1 и R2, чтобы на выходе получить (Vout) 14 В.

*Решение:*

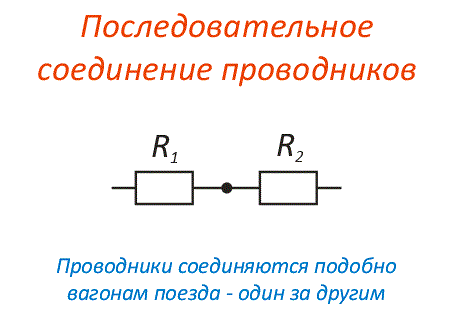
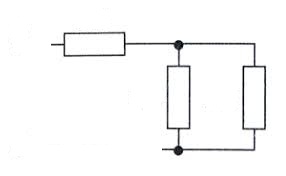
При :

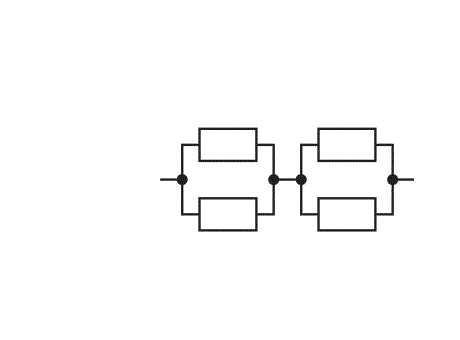
*Ответ:*R1 = 69, R2 = 15.

Реализация программы

В процессе работы над программой я решил переконфигурировать её для работы на основе телеграмм-бота. Я использовал библиотеку pytelegrambotapi (TeleBot). Данная библиотека полностью отражает функционал telegram api, с помощью неё можно отправлять фото, видео и реализовывать остальной функционал telegram.

Сообщения пользователей определяются с помощью декораторов (хандлеров) которые вызывают функцию ниже при получении сообщения определённого типа. Для того чтобы программа выполнялась в правильном порядке я использую флаги ожидания. Код программы есть в приложениях, вот его структура:

* + 1. Импорт библиотеки, создание объекта бота с токеном.
    2. Обработка команды /start, ожидание .txt файла с номиналами резисторов.
    3. Обработка полученного файла, поднятие флага на ожидание Vref.
    4. Обработка Vref, опускание флага Vref, поднятие флага на ожидание Vout.
    5. Обработка Vout, опускание флага.
    6. Запуск алгоритма:
  1.  Решение задачи с помощью двух резисторов.
  2. При отсутствии ответов к задаче, решать с помощью трёх резисторов (2 параллельно и один, и наоборот).
  3. При отсутствии ответов к задаче, решать с помощью четырёх резисторов.

1. Вывод ответа (при выполнении п. 6.3 также выводить погрешность).

Заключение

В данной работе была создана программа на языке Python 3.9, для работы телеграмм бота, выполняющая цель исследования: Создание компьютерной программы на языке программирования Python 3.9, которую можно будет использовать для решения задач по поиску резисторов для регулируемых источников питания.

При создании этой программы:

* Изучить научную литературу по данной проблеме;
* Найти и вывести формулы для расчёта сопротивлений резисторов;
* Изучить документацию к языку Python для поиска средств написания программы;
* Перевести алгоритм подбора по формулам на язык программирования Python.
* Создать интерфейс взаимодействия программы с пользователем;
* Добавить в код программы защиту от ошибок, неверного ввода чисел пользователем.

Я считаю, моего telegram-бота могут использовать люди, занимающаяся электроникой на предприятиях и обычные любители. Пользователи могут быстрее подбирать сопротивления, подходящие под их источник питания.

Список использованной литературы

1. Самоучитель по программированию на Python (https://pythonstart.ru/)
2. Документация по языку Python3 (https://docs-python.ru/)
3. Курс Основы Python (<https://ru.hexlet.io/courses/python-basics>)
4. Telegram api (<https://core.telegram.org>)
5. Операционный усилитель <https://habr.com/ru/companies/skillfactory/articles/658277/>

<http://www.unn.ru/books/met_files/OU_fin_crossword.pdf>

https://habr.com/ru/articles/508530/

1. <https://ru.wikipedia.org/wiki/Электрическое_сопротивление>

Приложения

* + - 1. Ссылка на Telegram-бот



* + - 1. Общий список резисторов

0.1 0.3 0.47 0.499 0.51 0.59 0.68 0.75 1 1.2 1.5 1.6 1.8 2 2.1 2.2 2.26 2.4 3 4.22 5.1 5.6 6.04 6.2 7.5 9.53 10 10.2 10.5 11.1 11.5 12 13.2 15 16 18 22 24 30 34 42.2 54.9 59 68 69 82 680

* + - 1. Текст программы

**from telebot import \***

**TOKEN = ‘Токен бота’ # @BotFather – генерирует токены управления**

**bot = TeleBot(TOKEN)**

**def File\_open(path, mode, write\_wa=None):**

**file = open(path, mode)**

**ret = file.read()**

**file.close()**

**return ret**

**def Calc\_paralel\_res(r1, r2):**

**r\_ecv = r1 \* r2 / (r1 + r2)**

**return r\_ecv**

**def Calc\_serial\_res(r2, v\_out, v\_ref):**

**r1 = r2 \* (v\_out/v\_ref - 1)**

**return r1**

**@bot.message\_handler(commands=['start'])**

**def start(message):**

**bot.flag\_waiting\_for\_v\_ref = False**

**bot.flag\_waiting\_for\_v\_out = False**

**bot.send\_message(message.chat.id, text="Загрузите файл со списком наминалов резисторов имеющихся в наличии:".format(message.from\_user))**

**@bot.message\_handler(content\_types=['document'])**

**def first(message):**

**try:**

**file\_info = bot.get\_file(message.document.file\_id)**

**downloaded\_file = bot.download\_file(file\_info.file\_path)**

**src = 'C:/Users/vlad/Documents/NPK2024/resistors\_tgbotV2/databot/' + message.document.file\_name**

**with open(src, 'wb') as new\_file:**

**new\_file.write(downloaded\_file)**

**bot.all\_resistorsT = File\_open(src)**

**bot.all\_resistorsLS = bot.all\_resistorsT.split()**

**bot.all\_resistors = list(map(float, bot.all\_resistorsLS))**

**bot.send\_message(message.chat.id, text="Введите Vref".format(message.from\_user))**

**bot.flag\_waiting\_for\_v\_ref = True**

**except:**

**bot.send\_message(message.chat.id, text="❌ Ошибка! Введите корректный файл .txt".format(message.from\_user))**

**return None**

**@bot.message\_handler(content\_types=['text'])**

**def second(message):**

**if bot.flag\_waiting\_for\_v\_ref == True:**

**try:**

**number = float(message.text.replace(',', '.'))**

**if number <= 0:**

**i = int('ERROR')**

**bot.send\_message(message.chat.id, f"✅ напряжение Vref = {number}В, Введите Vout".format(message.from\_user))**

**bot.v\_ref = number**

**bot.flag\_waiting\_for\_v\_ref = False**

**bot.flag\_waiting\_for\_v\_out = True**

**except (ValueError, AttributeError):**

**bot.send\_message(message.chat.id, "❌ Ошибка! Введите корректное число (например: 2.7, 1,9, не 0!)".format(message.from\_user)) # bot.register\_next\_step\_handler(message, process\_float\_input)  # Пов**

**return None**

**elif bot.flag\_waiting\_for\_v\_out == True:**

**try:**

**number = float(message.text.replace(',', '.'))**

**if number <= 0:**

**i = int('ERROR')**

**bot.send\_message(message.chat.id, f"✅ напряжение Vout = {number}В")**

**bot.v\_out = number**

**except (ValueError, AttributeError):**

**bot.send\_message(message.chat.id, "❌ Ошибка! Введите корректное число (например: 2.7, 1,9, не 0)")**

**return None**

**if bot.v\_out == bot.v\_ref:**

**bot.send\_message(message.chat.id, "R1 = 0, R2 = NC (Не подключено)".format(message.from\_user))**

**counter = 0**

**countersecond = 0**

**counterthird = 0**

**counterfourth = 0**

**for r2 in bot.all\_resistors:    #два резистора (--резистор--резистор--)**

**r1 = r2 \* (bot.v\_out/bot.v\_ref - 1)**

**if r1 in bot.all\_resistors:**

**resultat = 'R1 = ' + str(r1) + ' R2 = ' + str(r2)**

**bot.send\_message(message.chat.id, text=resultat.format(message.from\_user))**

**counter =+ 1**

**if counter == 0:    #три резистора (--=два резистора паралельно=--резистор--)**

**paralel\_var = {}**

**for r11 in bot.all\_resistors:**

**for r12 in bot.all\_resistors:**

**for r2 in bot.all\_resistors:**

**v\_out\_calc =  round(bot.v\_ref \* ((r11 \* r12) / ((r11 + r12) \* r2) + 1))**

**key = abs(bot.v\_out - v\_out\_calc)**

**paralel\_var.update({key: (r11, r12, r2)})**

**sorted\_paralel\_var = dict(sorted(paralel\_var.items()))**

**keys = list(sorted\_paralel\_var.keys())**

**result\_r = sorted\_paralel\_var[keys[0]]**

**# if bot.v\_out == v\_out\_calc:**

**bot.send\_message(message.chat.id, text=f'R1.1 = {result\_r[0]}, R1.2 = {result\_r[1]} (R1 эквивалент = {Calc\_paralel\_res(result\_r[0], result\_r[1])})\nR2 = {result\_r[2]}'.format(message.from\_user))**

**countersecond += 1**

**if countersecond == 0 and counter == 0:     #три резистора (--резистор--=два резистора паралельно=--)**

**paralel\_var = {}**

**for r1 in bot.all\_resistors:**

**for r21 in bot.all\_resistors:**

**for r22 in bot.all\_resistors:**

**v\_out\_calc =  round(bot.v\_ref \* ((r1 \* (r21 + r22)) / (r21 \* r22) + 1))**

**key = abs(bot.v\_out - v\_out\_calc)**

**paralel\_var.update({key: (r1, r21, r22)})**

**sorted\_paralel\_var = dict(sorted(paralel\_var.items()))**

**keys = list(sorted\_paralel\_var.keys())**

**result\_r = sorted\_paralel\_var[keys[0]]**

**# if bot.v\_out == v\_out\_calc:**

**bot.send\_message(message.chat.id, text=f'R1 = {result\_r[0]},\nR2.1 = {result\_r[1]}, R2.2 = {result\_r[2]} (R2 эквивалент = {Calc\_paralel\_res(result\_r[0], result\_r[1])})'.format(message.from\_user))**

**counterthird += 1**

**if countersecond == 0 and counter == 0 and counterthird == 0:   #четыре резистора (--два резистора паралельно--два резистора паралельно--)**

**paralel\_var = {}**

**for r11 in bot.all\_resistors:**

**for r12 in bot.all\_resistors:**

**for r21 in bot.all\_resistors:**

**for r22 in bot.all\_resistors:**

**v\_out\_calc =  round(((r11 / r21) \* (r12 / r22)  \* ((r21 + r22 ) / (r11 + r12)) + 1) \* bot.v\_ref, 2) # (((r11 \* r12) \* (r21 + r22) ) / ((r21 \* r22) \* (r11 + r12)) + 1) \* v\_ref**

**key = abs(bot.v\_out - v\_out\_calc)**

**paralel\_var.update({key: (r11, r12, r21, r22)})**

**sorted\_paralel\_var = dict(sorted(paralel\_var.items()))**

**keys = list(sorted\_paralel\_var.keys())**

**result\_r = sorted\_paralel\_var[keys[0]]**

**bot.send\_message(message.chat.id, text=f'Погрещность Vout: {keys[0]},\nR1.1 = {result\_r[0]}, R1.2 = {result\_r[1]} (R1 эквивалент = {Calc\_paralel\_res(result\_r[0], result\_r[1])})\nR2.1 = {result\_r[2]}, R2.2 = {result\_r[3]} (R2 эквивалент = {Calc\_paralel\_res(result\_r[2], result\_r[3])})'.format(message.from\_user))**

**bot.send\_message(message.chat.id, "Введите новое значение Vout для повторного вычисления, или напишите /start для запуска программы с начала".format(message.from\_user))**

**if \_\_name\_\_ == '\_\_main\_\_':**

**bot.polling()**