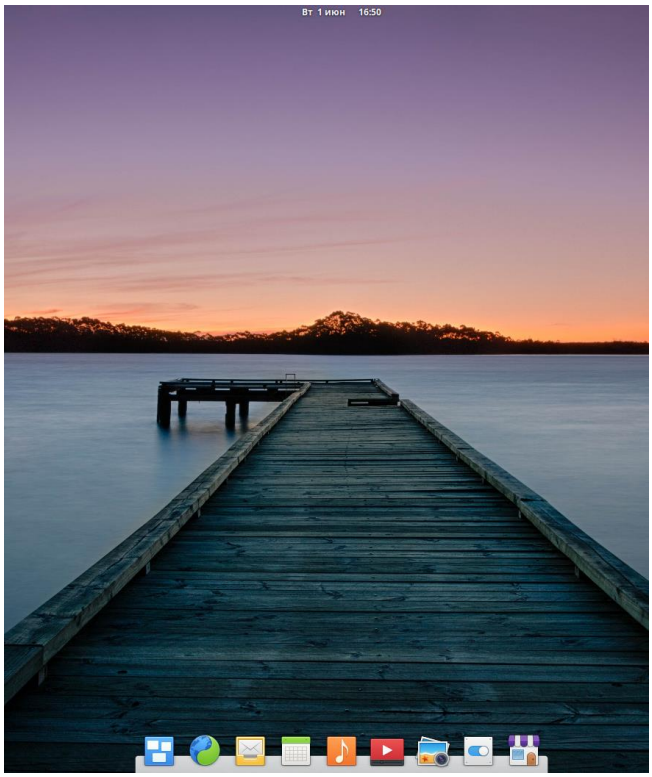


ELEMENTARYOS И ПРОГРАММЫ МОДЕЛИРОВАНИЯ

АННОТАЦИЯ

Радиолюбители зачастую интересуются программами, которые следует приобретать, но сделать это далеко не просто по разным причинам. Вместе с тем, или вместо того, можно использовать и бесплатную весьма красивую операционную систему, и полезные бесплатные программы для радиолюбителей. Начинающим бывает трудно ориентироваться при выборе как операционной системы, так и программ. Я надеюсь, что мой небольшой рассказ поможет им в этом.

Vladimir Gololobov

2021 г.

Оглавление

Глава 15. И это было давно (Oregano)

Завершался 2007 год. Пора, наверное, было устанавливать и наряжать ёлку, но позвонил главный редактор с просьбой назвать несколько вариантов названия книги, которую заказало издательство. Я был не готов назвать хотя бы одно, пока было только намерение рассказать о программах моделирования, с которыми я успел познакомиться. Бросив все дела, я стал придумывать названия, в чём преуспел не слишком, скажем прямо. У меня с названиями всегда плохо, что название книги, что название главы. Я понимаю, что издательству важно, чтобы каждое название вносило свой вклад в рекламу книги, но понимать, это одно, делать – совсем другое.

Почему это вспомнилось?

Время от времени у меня возникает желание посмотреть, что интересного есть в Linux. В этот раз желание навеяла статья в Яндекс-Дзен. Статья была краткой, но была фраза о скачивании дистрибутива, что напомнило мне – я уже пытался загрузить ElementaryOS, чтобы познакомиться с ней. Что же пошло не так в тот раз? Для скачивания предлагалось заплатить немного денег за операционную систему. Если бы я собирался использовать этот вариант Linux, я непременно заплатил бы, благо недорого, но любопытство не стоит даже небольших денег – не стоит его поощрять. Так что изменилось?

В статье было сказано, что можно ввести «0» в оплату за скачивание. Что я и сделал. От идеи устанавливать вторую ОС на компьютер я давно отказался в пользу использования VirtualBox от Oracle. После недолгих разборок с 32- и 64-битовой версиями ОС ElementaryOS стала устанавливаться. Установка мало отличается от аналогичных установок, где нужно указать местоположение, ввести пароль и т.д. Немного смутила скорость загрузки ОС: я привык в последних версиях Windows 10, что для загрузки достаточно нескольких секунд. Но это можно вполне списать на компьютер, на виртуальное представление и на то, что плохие привычки усваиваются быстро.

Неторопливая загрузка для меня вполне окупилась внешним видом интерфейса. Не знаю причины, но аналогичный интерфейс, кажется девятой версии, дистрибутива ROSA меня прельстил и ранее.

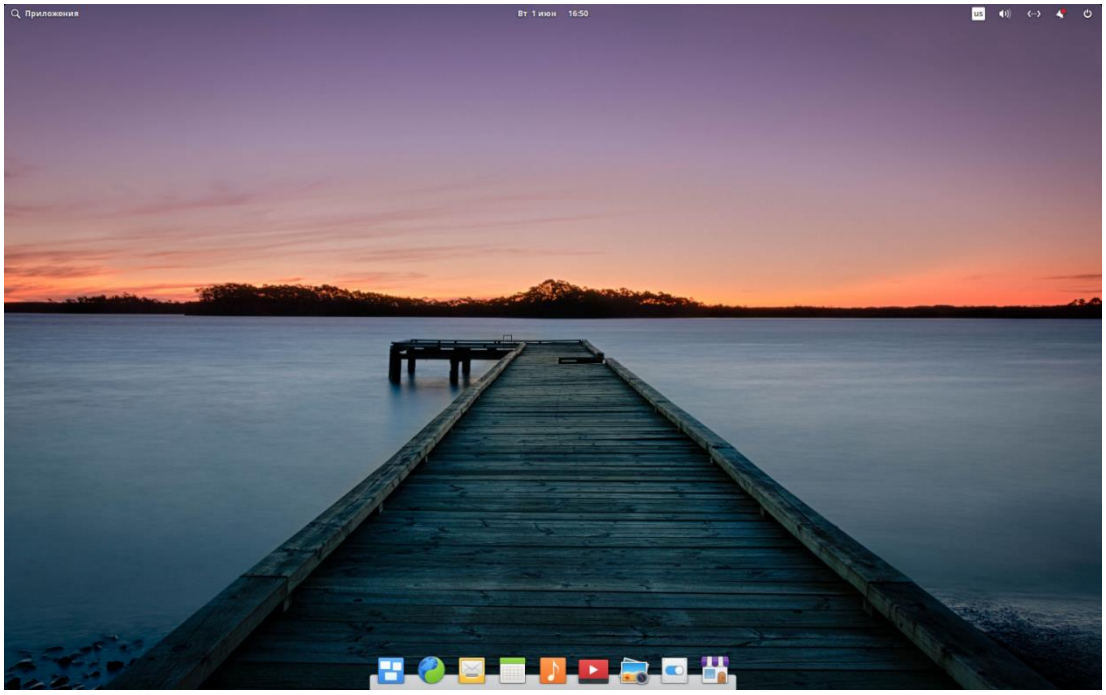


Рис. 15.1. Внешний вид загруженной ElementaryOS

Любая установка операционной системы начинается с ответов на вопросы, а завершается обращением к обновлениям. В данном случае можно поступить двояко – использовать терминал для обновления или использовать AppCenter (крайняя справа иконка на панели часто используемых приложений). Я использовал AppCenter, приложение показало все доступные обновления и предложило обновить выбранные позиции или всё сразу. Это доступно при выборе закладки «Установленные» (сейчас она показывает гораздо меньше).

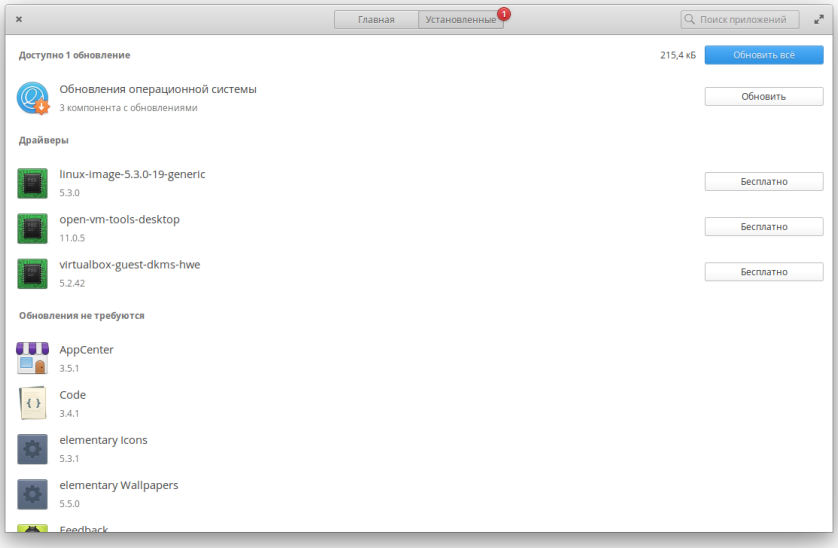


Рис. 15.2. Закладка «Установленные» в приложении менеджера установки приложений

Используя кнопку «Обновить всё» можно не задумываться о дальнейших шагах, а терпеливо дождаться завершения, процесс обновления отображается в окне приложений.

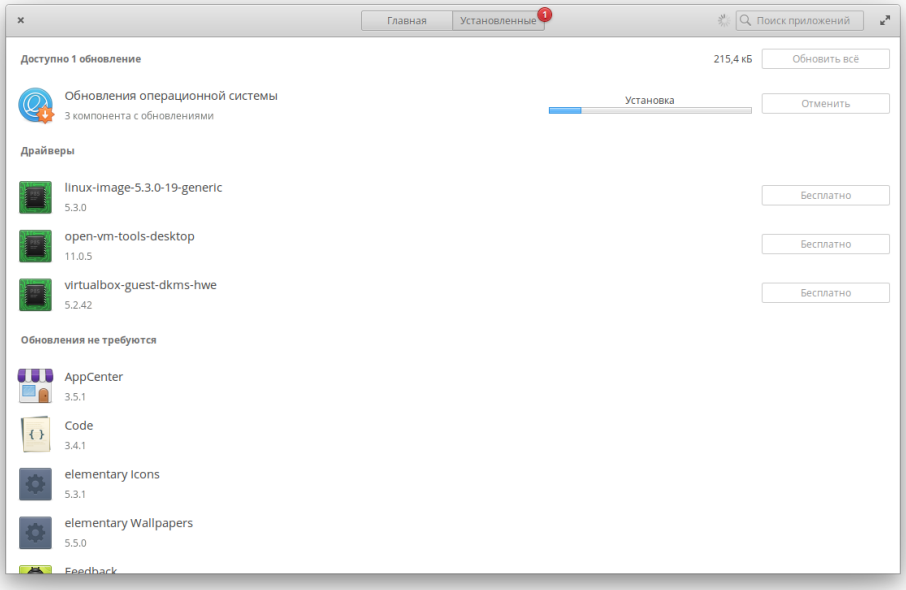


Рис. 15.3. Отображение процесса обновления

На верхней панели окна ОС в левом углу есть надпись «Приложения». Нажмите левой клавишей мышки, чтобы вывести все приложения доступные в данный момент. Например, можно выбрать файловый менеджер, двойной щелчок по иконке открывает окно менеджера. При этом на панели быстрого вызова (нижняя панелька) появится иконка менеджера файлов. По ней можно щёлкнуть правой клавишей мышки, чтобы прикрепить и это приложение к панели быстрого доступа к приложениям. В нижней части окна приложений есть переключатель страниц.

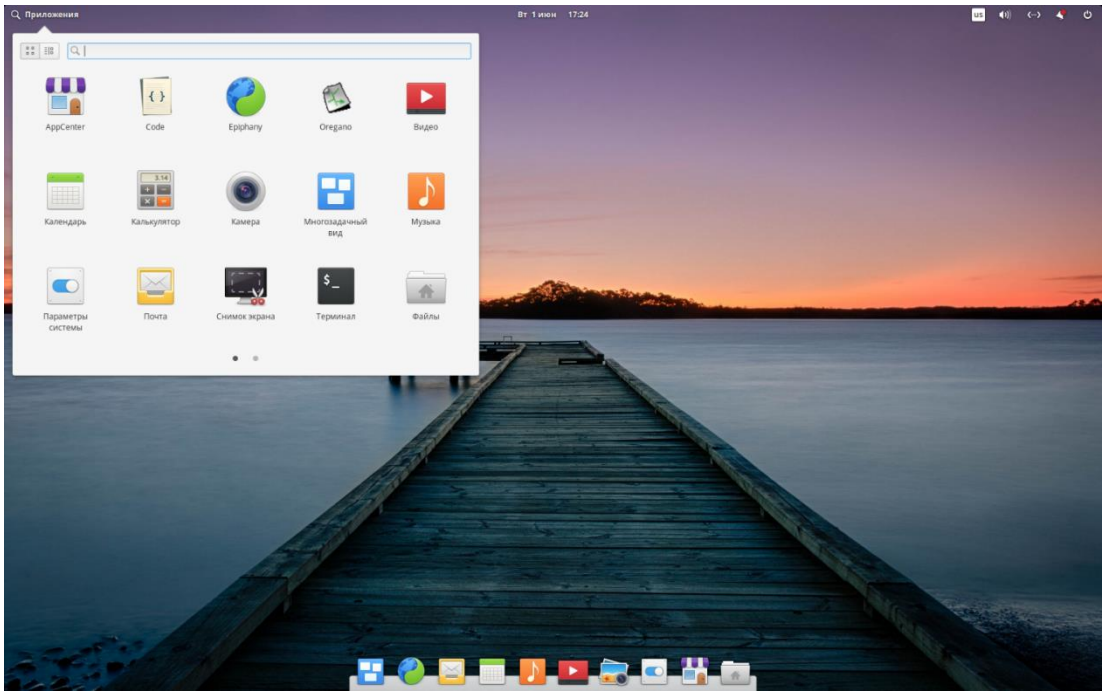


Рис. 15.4. Приложения в ElementaryOS

Пока не забыл, после установки пришлось зайти в настройки (параметры системы, иконка с выключателем), чтобы выбрать русский язык системы. После загрузки необходимого к смене языка достаточно выйти из системы, но лучше, возможно, перезагрузить систему.

Вот мы и подошли к тому, что имеет непосредственное отношение к рассказу. Если вас интересует просмотр видео, посещение социальных сетей и копирование фотографий с телефона на компьютер, вам достаточно того набора приложений, что подготовили разработчики дистрибутива. Но меня интересовали программы для радиолюбителей. Их, используя AppCenter (менеджер приложений), можно поискать, например, в разделах «Математика, наука и инженерия» и «Образование». Оба раздела весьма, следует признать, представительны.

Там-то я и наткнулся на программу Oregano, с которой бегло познакомился много лет назад. Позже я пытался найти её на просторах Интернета, но безуспешно. А сегодня, по прошествии стольких лет, я и не вспомню, как работать с программой. Придётся напомнить.

Регенеративный приемник. Oregano, Linux (2007 г.)

Из одного письма:

> *Собрал я регенеративный приемник (вот ссылка) http://picbasic.net.ru/shems/reg-kvp.htm, звук слышен даже очень хорошо только через 6 секунды он плавно начинает уменьшатся (уменьшается он секунд 8), не слышно его секунды 3, а потом он (звук) начинает также плавно, как и опускался, подниматься, и так на протяжении всего прослушивания. Подскажите, пожалуйста, в чем неисправность. В деталях? В контактах? И вот еще, где*

можно найти подробное описание работы этого приемника (статью Полякова), где можно найти усилитель, чтобы можно было слушать не через головные телефоны, а через динамик (обычный усилитель как я понимаю не подойдет). Настоятельно прошу помочь в первой проблеме, так как с таким звуком мне не будет нужен и усилитель, и описание такого приемника.

Помню в юные годы я пытался собрать приемник прямого усиления из конструктора. Достаточно безуспешно. Не хотел он работать так, как мне хотелось бы. Позже с просьбой помочь в «оживлении» приемника обращались ко мне «измученные» родители, и не слишком я им помог. Можно сразу сослаться на то, что я не специализируюсь в области именно радиотехники, но мне сегодня интересно, чем могут помочь в этом вопросе программы EDA. Отправившись по ссылке, я «срисовываю» схему приемника, и «навскидку» мне приходит в голову, что управление усилением по постоянному току, проходящее по цепи R6C2R1, может зависеть от характера исходного сигнала и колебаться при неблагоприятной ситуации ровно так, как описывает автор. Но очень хотелось бы проверить свои соображения. Для начала приведу схему приемника:

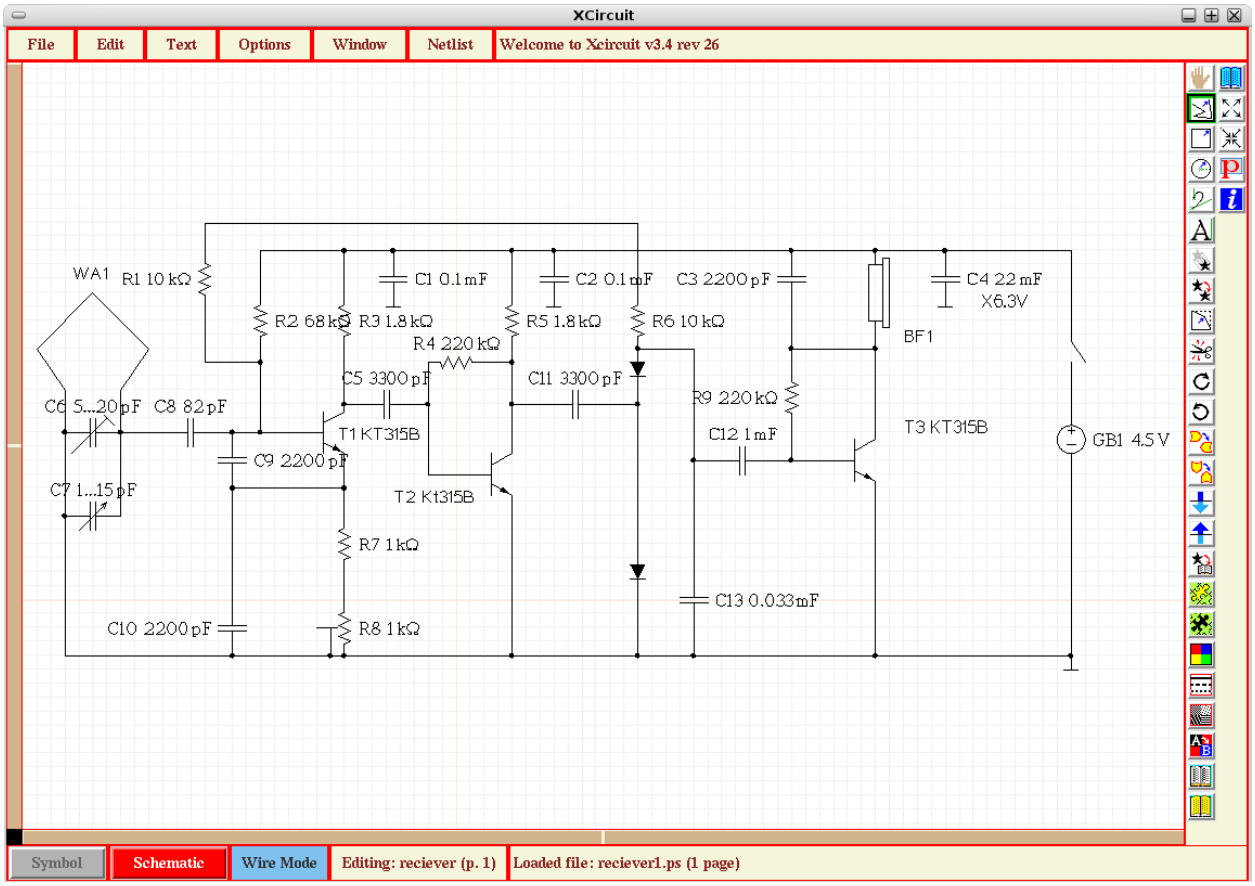


Рис. 15.5. Схема регенеративного приемника (оригинал)

Первое, что я хочу попробовать, это поработать со схемой в программе Oregano. Программа, как мне кажется, весьма удобна для начинающих любителей электроники, и работа с ней в Linux мало чем отличается от работы с такими программами в Windows как Electronics Workbench. Как запустить программу и начать в ней рисовать схему, я уже писал в обзоре, но повторю отдельные моменты. Запускаем программу и получаем возможность сразу расставить в окне редактора компоненты, которые берем из правого окошка менеджера компонентов. Для начала, я думаю, достаточно тех, что есть в библиотеке Default:

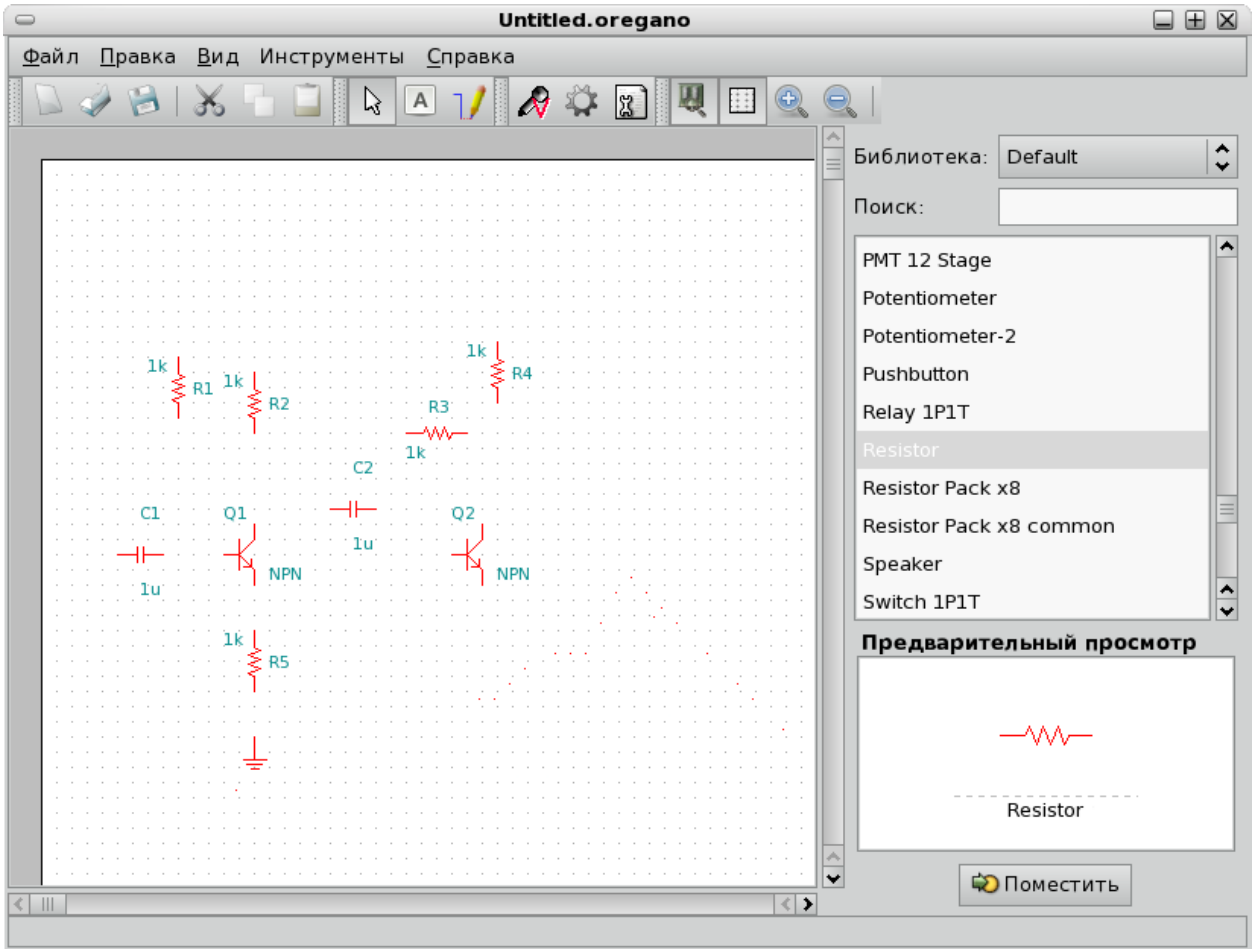


Рис. 15.6. Окно редактора в программе Oregano и окно компонентов

При размещении компонент можно выделить и, щелкнув по нему правой клавишей мышки, в выпадающем меню выбрать пункт «Повернуть». А нажав на иконку с изображением карандаша на инструментальной панели, провести нужные соединения, чтобы получить схему. Пока я не рисую цепь обратной связи по постоянному току, чтобы проверить, а работает ли моя схема без нее. Двойным щелчком по компонентам я открываю диалоговое окно свойств компонента, в котором меняю значения резисторов и конденсаторов, и задаю напряжение и частоту источника сигнала, который находится в самом низу библиотеки под именем VSIN. Попутно я сталкиваюсь с тем, что диоды детектора, которые я добавил в схему, 1N4148 не имеют модели, что не позволяет запустить моделирование. Посмотрев, согласно с появившимся сообщением об ошибке, нужную папку я обнаруживаю, что есть модель диода 1N750, которую и прописываю в свойствах компонента. Позже можно будет посмотреть – если файл модели имеет формат простого текстового файла, то можно будет добавить модель и этого диода. Дополнительно, выбрав на инструментальной панели иконку с изображением пробника, я устанавливаю два пробника: первый на генератор, второй на выход усилителя.

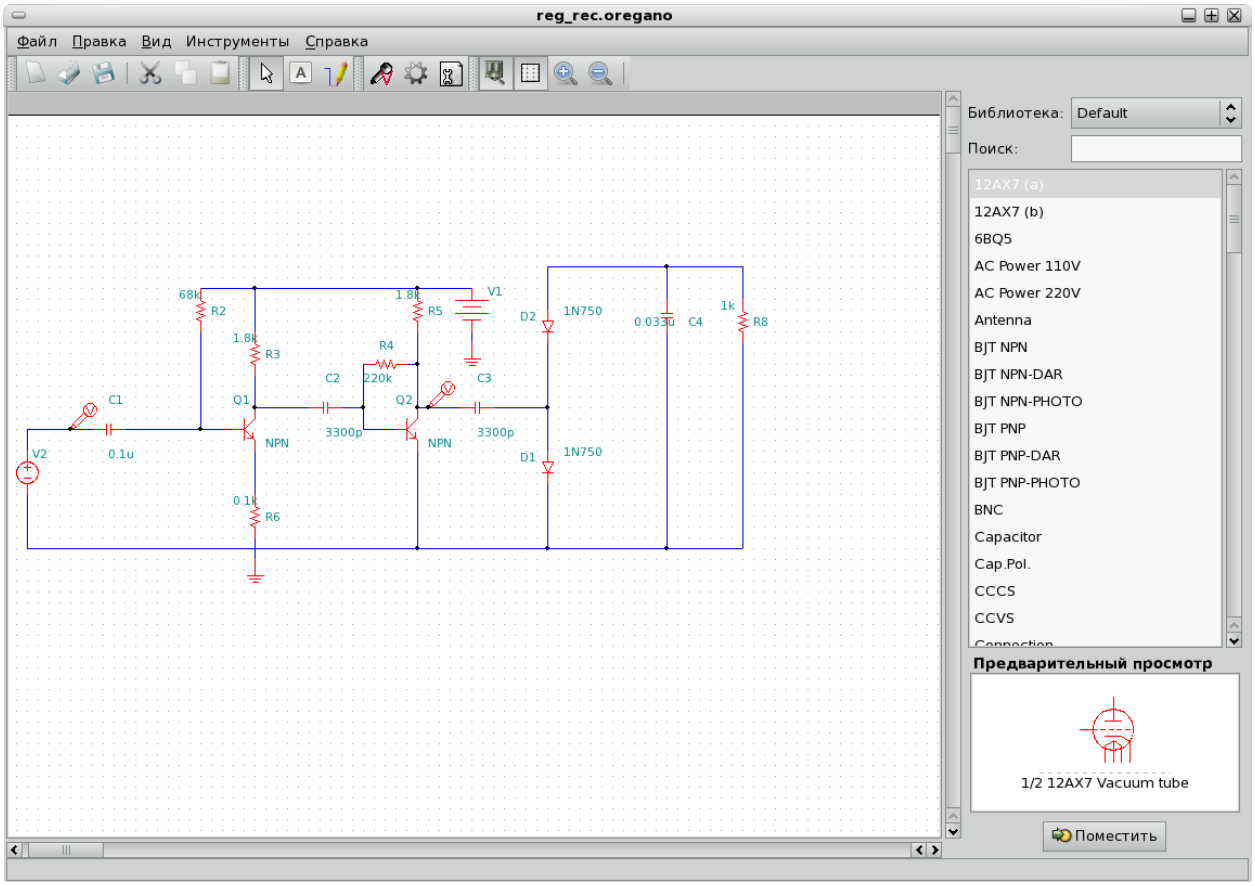


Рис. 15.7. Схема радиоприемника в программе Oregano

Теперь, когда схема нарисована, значения резисторов и конденсаторов (кроме C1, который я намерено увеличил до 0.1мкФ) соответствуют оригинальной схеме, я хочу посмотреть, как выглядели бы мои сигналы на экране осциллографа, если бы я собрал схему на макетной плате. Для этого я выбираю в основном меню «Инструменты-Моделировать», и...

В открывающемся окне графики слева есть раздел Nodes-Узлы. Если раскрыть этот пункт меню графического окна, то в нем обнаруживаются два моих пробника под именами V(8) и V(3), что свидетельствует о том, что это не первая моя попытка получить необходимое изображение.

И действительно, я уже побывал в разделах основного меню «Настройки моделирования» и «Параметры», которые позволяют выбрать режимы работы моделирования (и для симуляции в программе выбрать подходящий симулятор) в окнах диалога, открывающихся при выборе этих пунктов меню. Причина этого интереса в том, что не вижу я сигналов в окне графики.

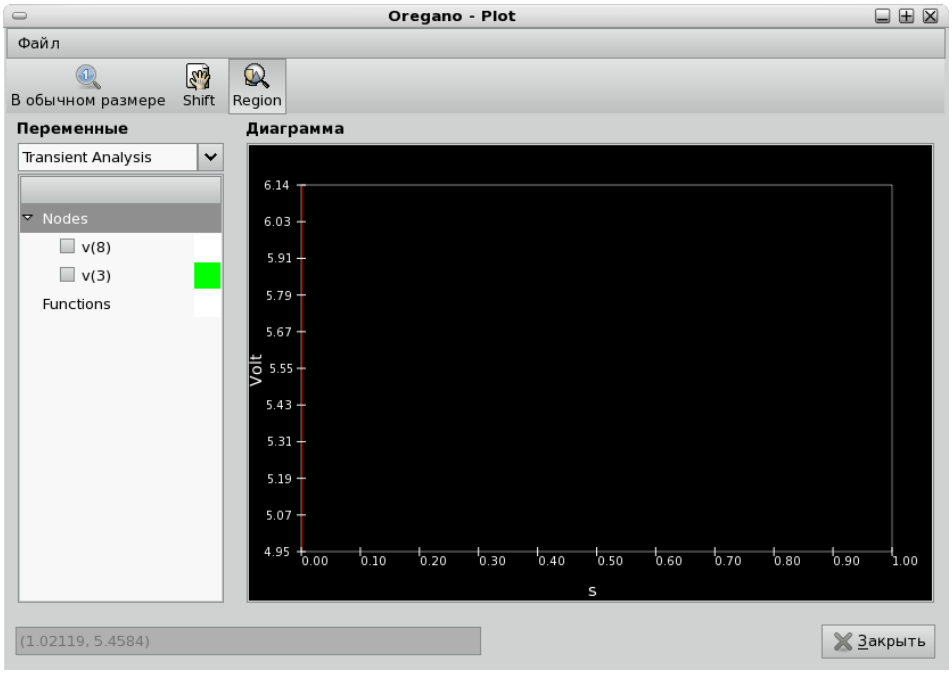


Рис. 15.8. Окно графического результата моделирования в программе Oregano

Прежде чем отказаться от дальнейших попыток, а у меня есть намерение перейти в случае неудачи к работе с другой программой, я, пожалуй, поменяю задачу. Упрощу ее до предела. Возьму два резистора, генератор и попробую посмотреть, что же у меня получается (или не получается). Все значения и настройки я пока оставлю без изменения.

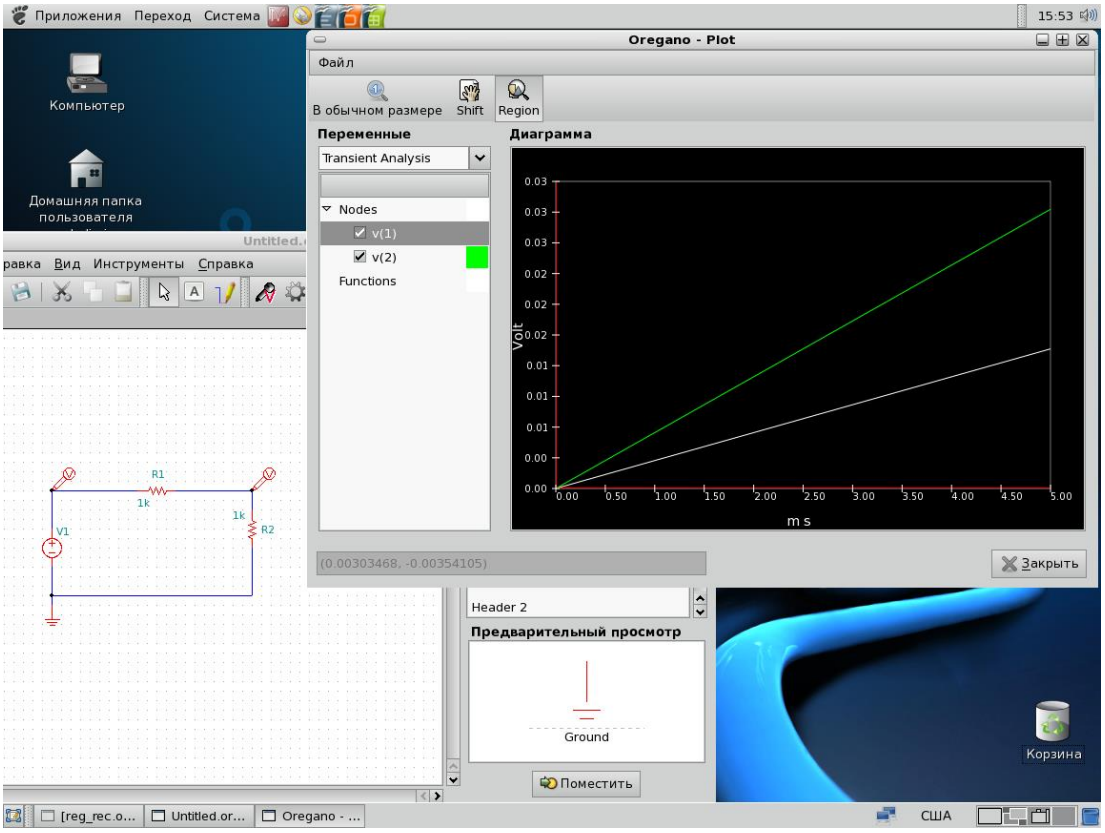


Рис. 15.9. Простейшая схема в программе Oregano

Получившиеся две прямые явно не то, что я хотел бы увидеть. Но в свойствах генератора в окошке частота я вижу значение 1.0. Возможно, частота генератора 1 Гц, в то время как на графике последнее значение отвечает 5 мС. Попробую я увеличить значение частоты генератора до 100.0. В результате получается следующий график:

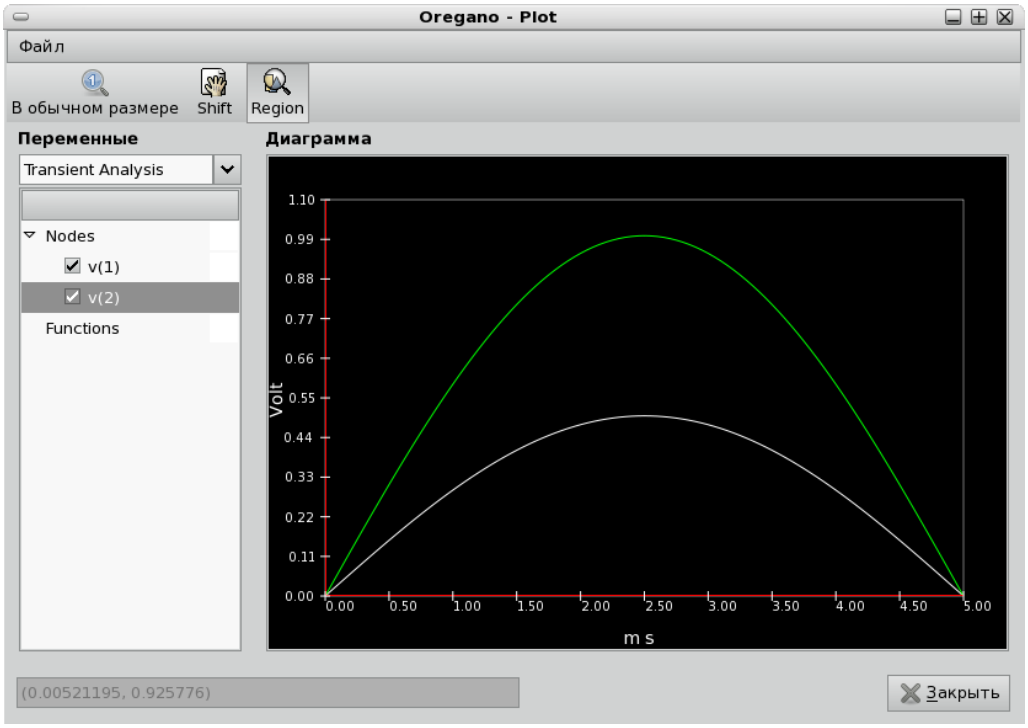


Рис. 15.10. График при увеличении частоты генератора в 100 раз

Это больше напоминает мне искомый вид графика. Но теперь я хочу заменить резистор R1 конденсатором и увеличить частоту до 10000.0.

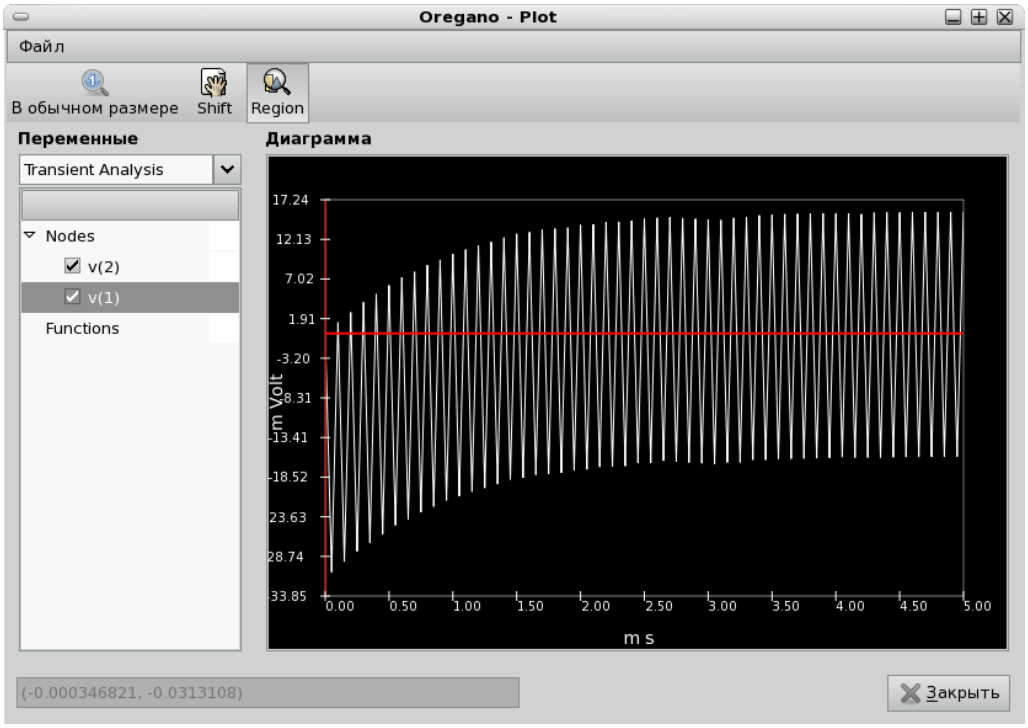


Рис. 15.11. График при замене резистора конденсатором и увеличении частоты

Если уменьшить частоту в десять раз, или в «Настройках моделирования» изменить значение 5 мС на 0.5 мС (ms, соответственно), то «картинка» меня вполне удовлетворяет. Попробуем добавить транзистор к схеме. Конечно, кроме транзистора нужно добавить еще резисторы, источник питания транзистора, изменить номиналы резисторов и источника питания (VDC, где я устанавливаю значение 4.5).

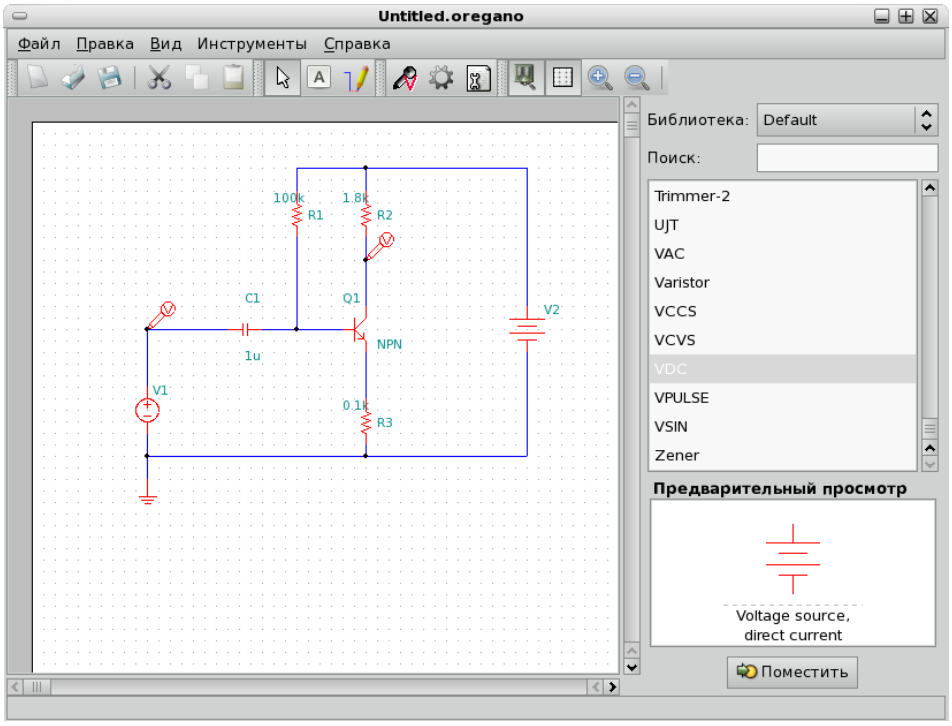


Рис. 15.12. Усложнение схемы в программе Oregano

И график сигналов показывает мне, что значения резисторов я выбрал «наугад», но не вполне удачно.

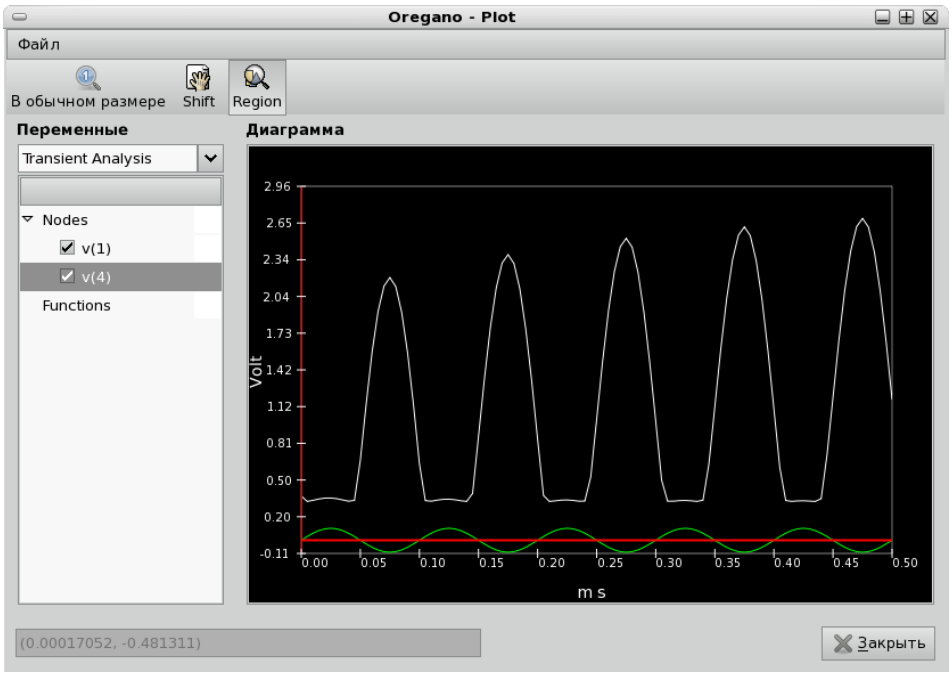


Рис. 15.13. Сигналы генератора и усилителя в предыдущей схеме

Верхний (выходной) сигнал явно показывает ограничение сигнала снизу. Но, с другой стороны, если обратиться к предыдущему графику, то можно отметить, что на диаграмме отображается и начальная стадия процесса. Таким образом, возможно, что в данном случае нормальный режим работы еще не установился. Добавим еще один транзистор с соответствующими компонентами, немного поправим настройки, чтобы получить нечто похожее на задуманное.

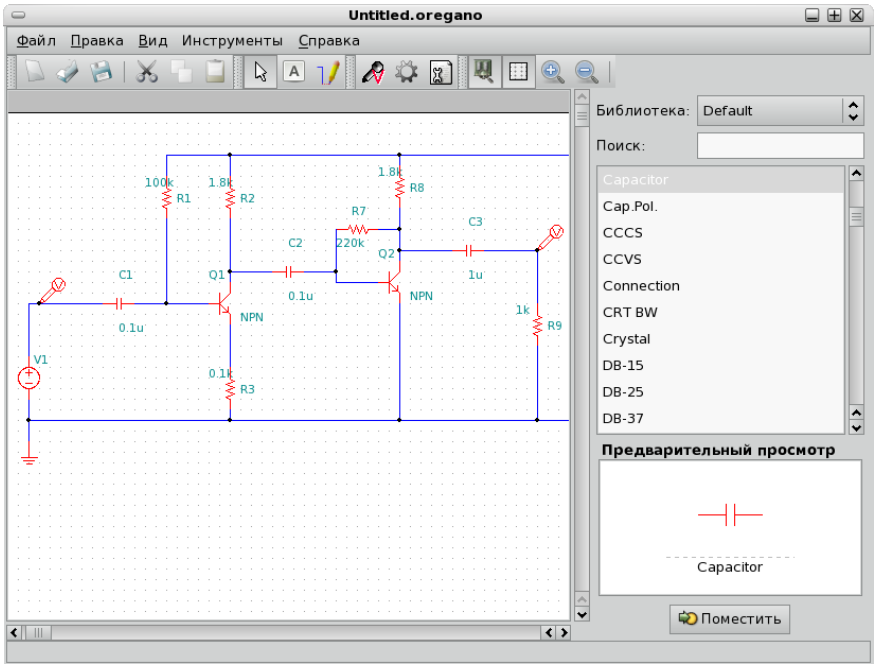


Рис. 15.14. Схема в программе Oregano близкая к оригинальной схеме приемника

Эту схему отличает отсутствие детектора, но сигнал на выходе усилителя уже совсем похож на «настоящий».

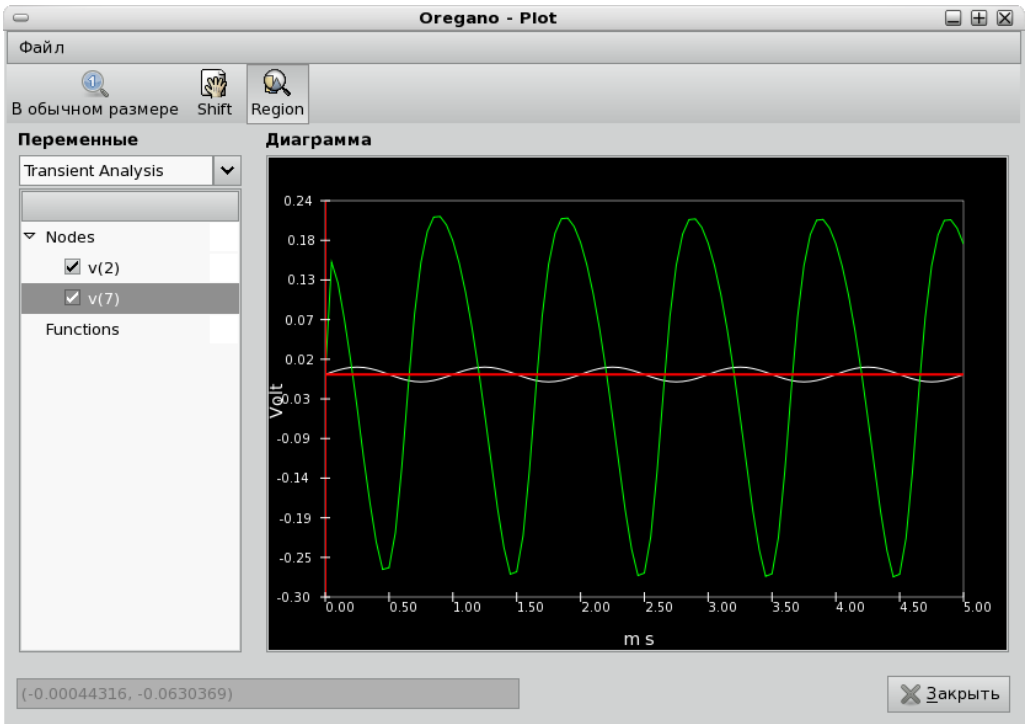


Рис.15. 15. Сигналы на выходе и входе усилителя в предыдущей схеме

Здесь меньший сигнал – V(2), относится к генератору. Мне кажется, что добавление детектора не должно сказаться на работе программы. Но, чтобы проверить это, надо добавить детектор. И убедиться, что все работает при соответствующих настройках. А теперь я хочу показать, какое предположение я хотел иллюстрировать с помощью программы Oregano. Для этого я добавлю второй генератор, частоту которого я выберу меньше частоты первого генератора. Схема эксперимента получается следующей:

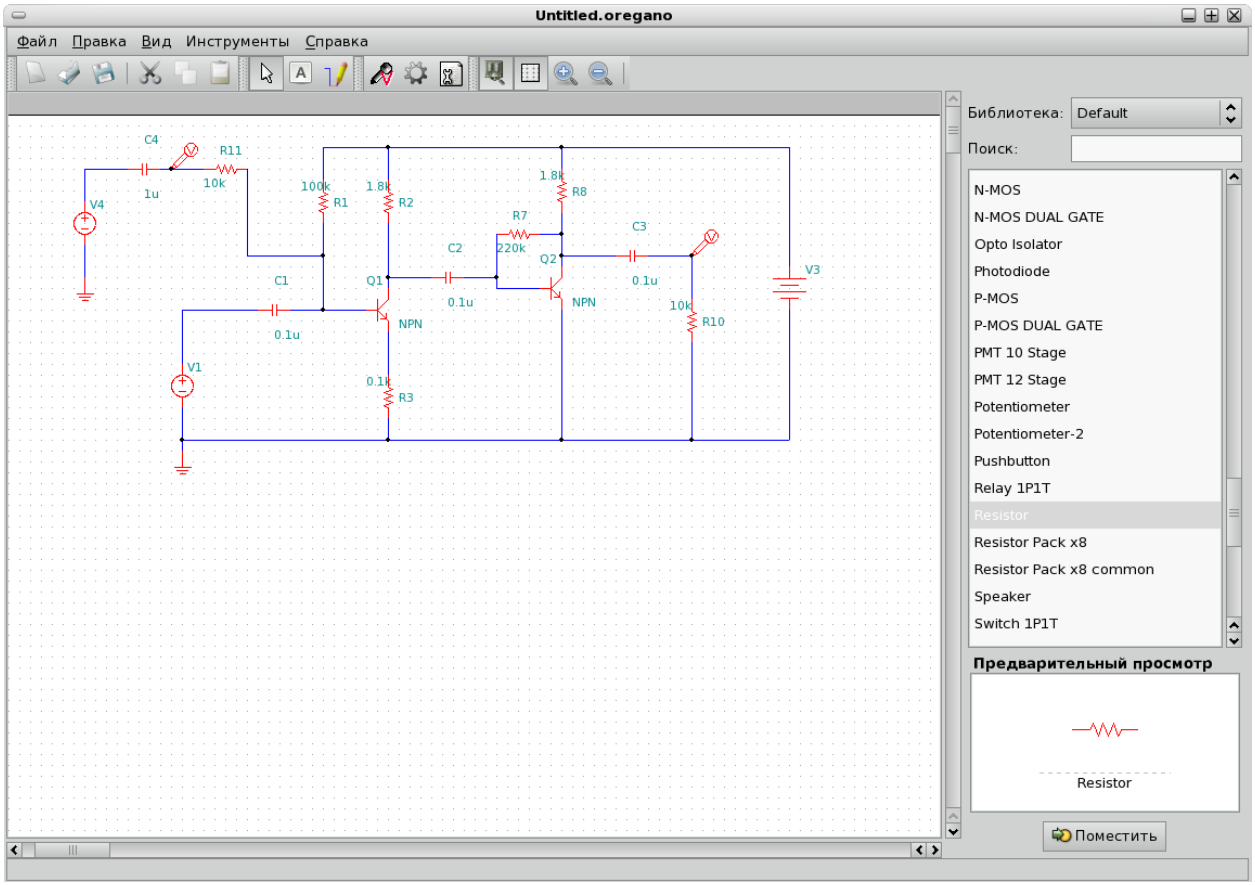


Рис. 15.16. Схема эксперимента с двумя генераторами

В моем эксперименте второй генератор, верхний по схеме, играет роль управляющего сигнала, заменяющего обратную связь через цепь R6C2R1 в оригинальной схеме. И я хочу проверить предположение, что при неблагоприятных условиях эта обратная связь может управлять сигналом, приходящим с антенны на вход приемника. Моделирование схемы эксперимента дает следующую картинку:

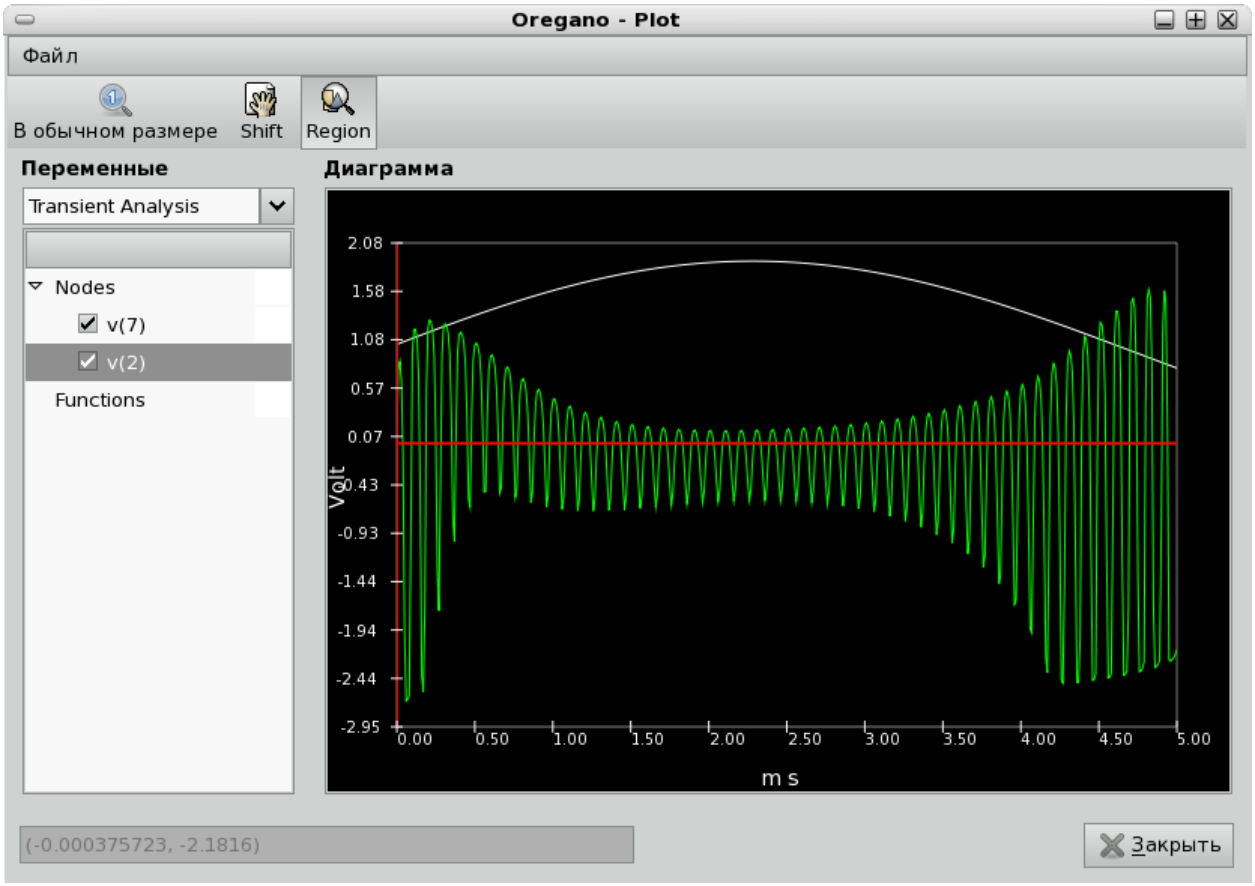


Рис. 15.17. Вид сигналов от второго генератора и на выходе усилителя

Верхний график – это сигнал от второго генератора, который явно изменяет вид выходного сигнала. Последний соответствует сигналу несущей частоты радиостанции, на которую настроен приемник. Если «нечто» работает подобным же образом, то уменьшение сигнала несущей частоты может в конечном счете «прослушиваться», как затихание звука. Что может быть этим «нечто»? Не знаю. Я могу предположить, что причина в неудачно выбранной постоянной времени цепи R6C2R1, или замене рекомендованного транзистора транзистором другого типа. Разные типы транзисторов могут по-разному реагировать на изменение тока базы транзистора. Можно предположить и наличие некоторой «мощной» помехи в месте проверки приемника. Помехи высокой частоты, которая может «нагревать» транзистор, смещая его режим работы в сторону уменьшения усиления (минуя цепь обратной связи), но затем цепь обратной связи восстанавливает режим работы. В реальной схеме в месте проверки я бы посмотрел, поскольку сигнал меняется очень медленно, не меняется ли напряжение (по постоянному току) в точке соединения резисторов R6 и R1. Словом, продолжил бы поиск причин этого явления. Но меня сейчас больше интересует программа Oregano, к ней я и вернусь.

Я хочу посмотреть, действительно ли файл модели диода 1N750, который у меня есть, текстовый файл. Я нахожу его в директории /usr/local/share/oregano/models. Он открывается простым текстовым редактором и выглядит, похоже, как текстовый файл.

```
.model 1N750 D(Is=880.5E-18 Rs=.25 Ikf=0 N=1 Xti=3 Eg=1.11 Cjo=175p
M=.5516
+  Vj=.75 Fc=.5 Isr=1.859n Nr=2 Bv=4.7 Ibv=20.245m Nbv=1.6989
+ Ibvl=1.9556m Nbvl=14.976 Tbvl=-21.277u)
```

Если так, то можно будет добавлять свои модели в программу, что значительно расширяет возможности ее использования. Если программу в том виде, в каком она есть, можно рекомендовать начинающим радиолюбителям, то опытные радиолюбители могут получить от нее гораздо больше. Гораздо больше получат и начинающие радиолюбители, если будут достаточно настойчивы и любознательны (а не любопытны, как я). Вот еще один вид графического окна той же схемы.

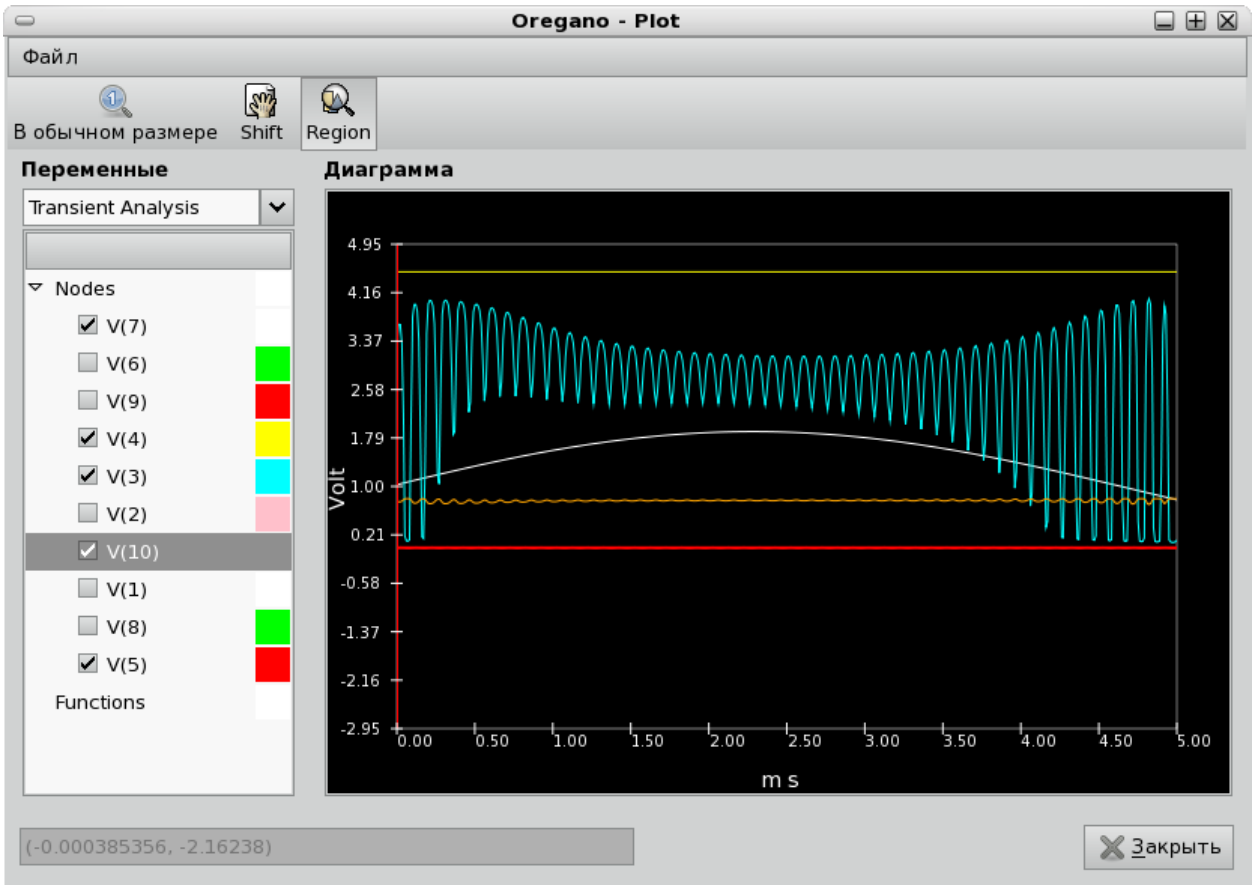


Рис. 15.18. Графическое окно при использовании симулятора NgSpice

Это изменение произошло после того, как в разделе «Параметры» я выбрал Механизм-NgSpice на основной вкладке «Пользовательский интерфейс». Конечно, программу ngspice я своевременно установил на компьютер.

Обширная библиотека «по умолчанию» программы Oregano и наличие еще нескольких библиотек навело меня на мысль о проведения разного рода интересных экспериментов. Но, увы, радость моя была преждевременной. Кроме базовых элементов программа, возможно, работает с еще несколькими дополнительными моделями, и я даже не исключаю, что при достаточной настойчивости в освоении ngspice, поискав существующие модели, можно существенно расширить возможности программы. Но лучше подождать, когда новая версия программы уже будет пополнена в части используемых моделей. Полезность же дополнительных моделей я могу проиллюстрировать в настоящий момент только одним примером.

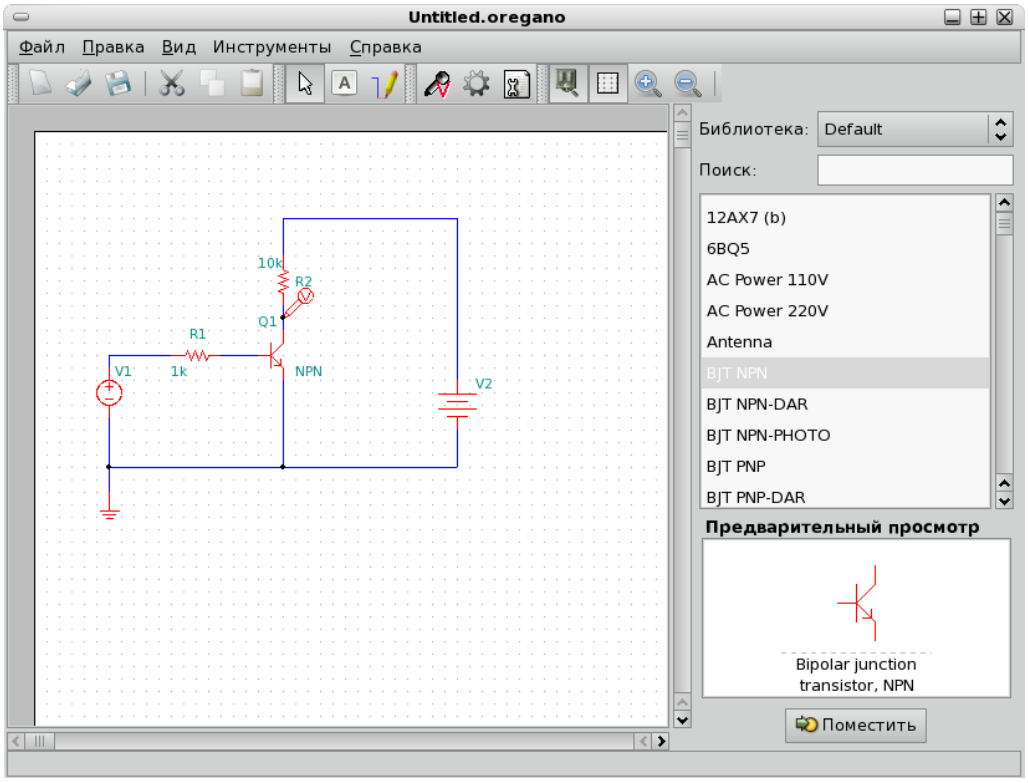


Рис. 15.19. Схема с идеальным транзистором в программе Oregano

Используемый источник питания, естественно, я перестраиваю на 10.0 (вольт), а источник сигнала на 1000.0 (1000 Гц). Если теперь запустить моделирование, то в графическом окне можно увидеть результат.

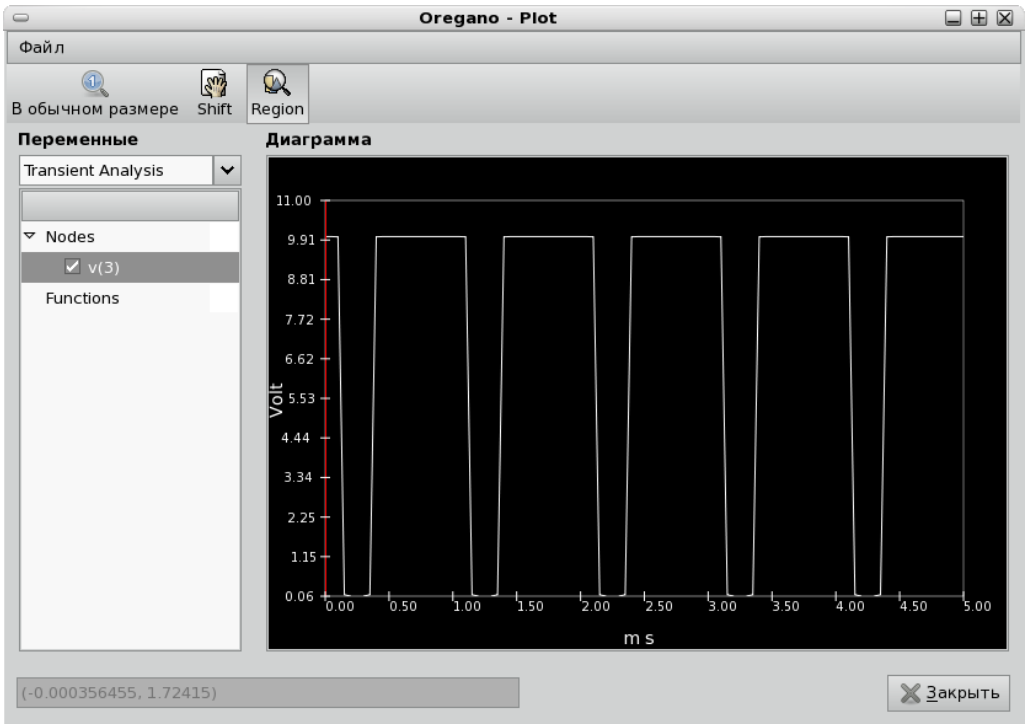


Рис. 15.20. График работы идеального транзистора

Теперь, изменив в свойствах транзистора идеальную модель NPN на реальную модель 2N2222, и повторив моделирование, мы получим несколько иную картину.

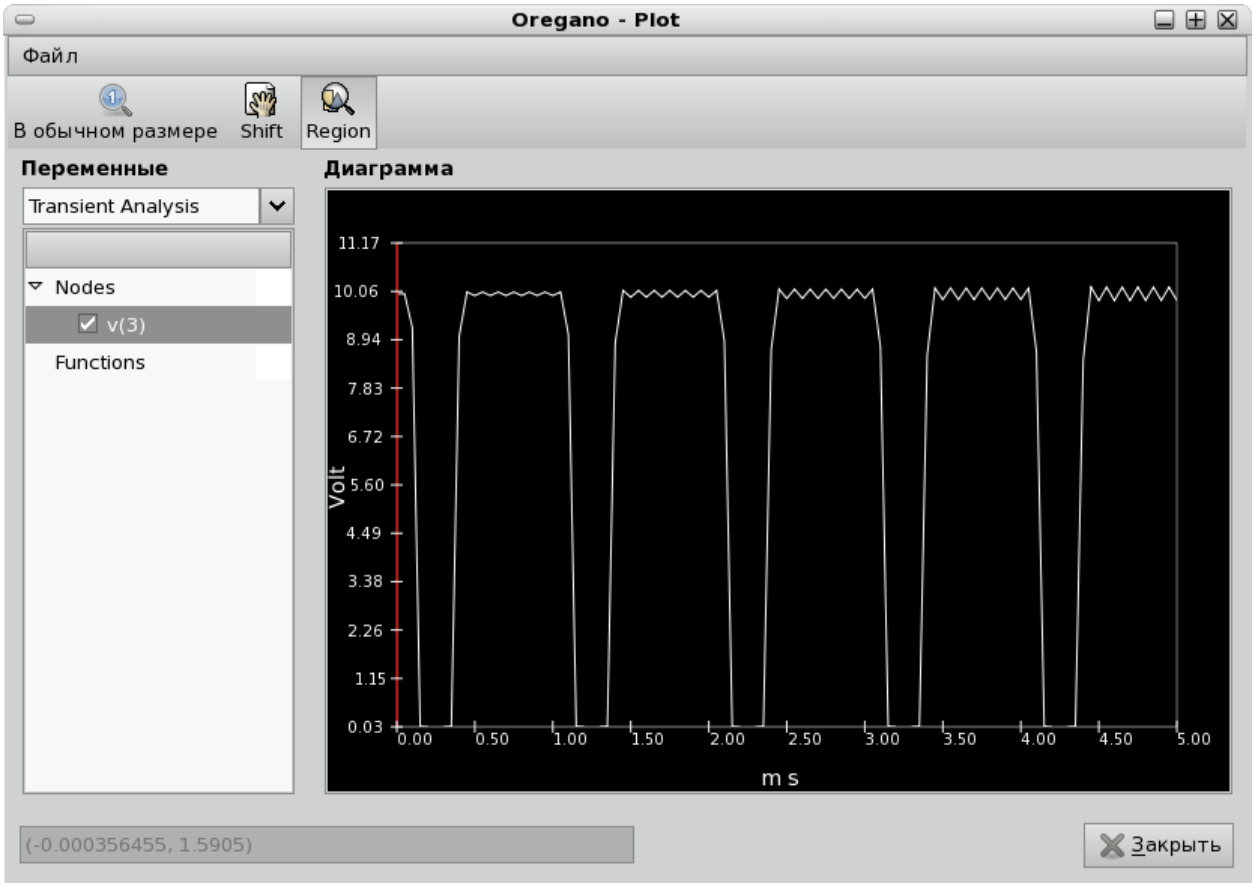


Рис. 15.21. График работы реального транзистора

Оба графика очень похожи, если не обращать внимание на вершину импульсов, полученных из синусоидального сигнала. Я думаю, в реальном транзисторе отображаются шумы транзистора, обязанные своим появлением, судя по изменению их амплитуды, паразитным емкостным и индуктивным свойствам реального транзистора.

В попытках рассказать много интересного из интимной жизни микросхем я, сознаюсь, потерпел фиаско. И что может меня оправдать?

Во-первых, даже с уже работающими компонентами можно провести очень много экспериментов, особенно начинающим радиолюбителям, для которых простота работы с программой гораздо важнее, чем тонкости моделирования.

Во-вторых, я честно потратил весь выходной день на поиски документации по моделированию, поиск моделей. Нашел модели, которые должны были работать с симулятором, стимулированный заменой одного из транзисторов на найденную модель, я добавил в папку /usr/local/share/oregano/models множество моделей, исправляя их расширения с .mod на .model, загрузил объемистое руководство по ngspice (более 200 страниц, из которых прочитал, если честно, не более двух десятков), и даже в лекция профессора университета Беркли нашел рассказ о моделях в Spice, который тоже прочитал. Я славно, хотя и без пользы, потрудился, что несколько оправдывает, надеюсь, меня в ваших глазах. А что до неудач, я к ним привык, главное, чтобы они не останавливали. Можно отложить решение проблемы, это даже полезно, если множество попыток не дает положительного результата, но не бросать всё раз и навсегда.

Oregano, ElementaryOS (2021г.)

Вспоминая работу с программой Oregano, я понял, что некоторые аспекты описаны настолько «вскользь», что понять, как это осуществить на практике, не представляется возможным. Например, в программе есть модель идеального транзистора, но я использовал транзистор 2N2222. Как?

Попробуем разобраться. Итак. Я нашёл текстовое описание модели этого транзистора (многие, похоже, модели Spice такие же):

```
.model 2N2222 NPN(IS=1E-14 VAF=100
+    BF=200 IKF=0.3 XTB=1.5 BR=3
+    CJC=8E-12 CJE=25E-12 TR=100E-9 TF=400E-12
+   ITF=1 VTF=2 XTF=3 RB=10 RC=3 RE=1 Vceo=30 Icrating=800m  mfg=Philips)
```

Расширение файла было .mod, что легко исправилось на .model. Этот файл следовало перенести по адресу: /usr/share/oregano/models. В разных дистрибутивах с этим возникает проблема нехватки прав на доступ к системным директориям. В ElementaryOS проблема решилась просто: достаточно щёлкнуть правой клавишей мышки по диспетчеру файлов (проводнику), чтобы выбрать раздел запуска от имени администратора. Файл оказался в нужном месте, но в окне выбора компонентов я не увидел транзистора 2N2222. Что дальше?

А дальше всё оказалось достаточно прозаично – добавляем идеальный транзистор в окно редактора, открываем его свойства (щелчок правой клавишей мышки по компоненту, выбор Object Properties). Вместо модели NPN вписываем наш транзистор 2n2222.

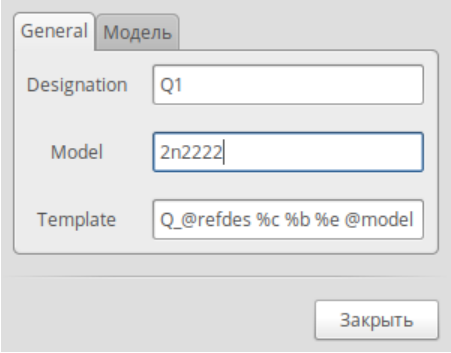


Рис. 15.22. Замена идеального транзистора на реальный

Осталось нажать кнопку «Закрыть», хотя можно увидеть на закладке «Модель» описание модели.

Есть одно замечание по работе ОС в VirtualBox (полновесную работу я проверить не могу). После загрузки можно долгое время наблюдать чёрный экран. В полноэкранном режиме это непонятно, настораживает, стимулирует к действиям. Но в обычном виде можно наблюдать, как работает виртуальный жёсткий диск, как идёт обращение к сети – операционная система живёт своей жизнью, ей явно не до пользователя.

Чтобы использовать симулятор ngspice достаточно в терминале дать команду:

```
sudo apt-get install ngspice
```

Программа загрузится, установится, и её можно использовать.

Первая попытка отобразить усиление транзистором в однокаскадном включении по схеме с общим эмиттером меня несколько насторожила, усиление каскада по напряжению порядка двух. Маловато будет. А АЧХ (режим АС) выводится прямой линией на графике. Хотя, если транзистор идеальный, его АЧХ может быть и такой.

К слову, в дистрибутиве ElementaryOS работает и клавиша Print Screen, и Alt+Print Screen, что мне, скажем, привычнее для получения снимков экрана. Найти их несложно: папка «Изображения», далее папка снимков экрана (Screenshots). Но вернёмся к программе.

Итак. У нас модель реального транзистора. Посмотрим его АЧХ: основное меню Edit, Настройки моделирования. Снимаем «галочку» в разделе «Переходный», ставим её в АС, где оставим всё пока без изменений, кроме добавления предела частоты, увеличив её до 1000 Meg. В настройках источника напряжения в разделе АС укажем напряжение в 1 В. И…

И получим ровную линию графика АЧХ вплоть до 1 ГГц. Что-то с моделью реального транзистора, прямо скажем, не задалось. Хотя… может быть, не работает анализ на переменном токе? Проверим. Создадим новую схему.

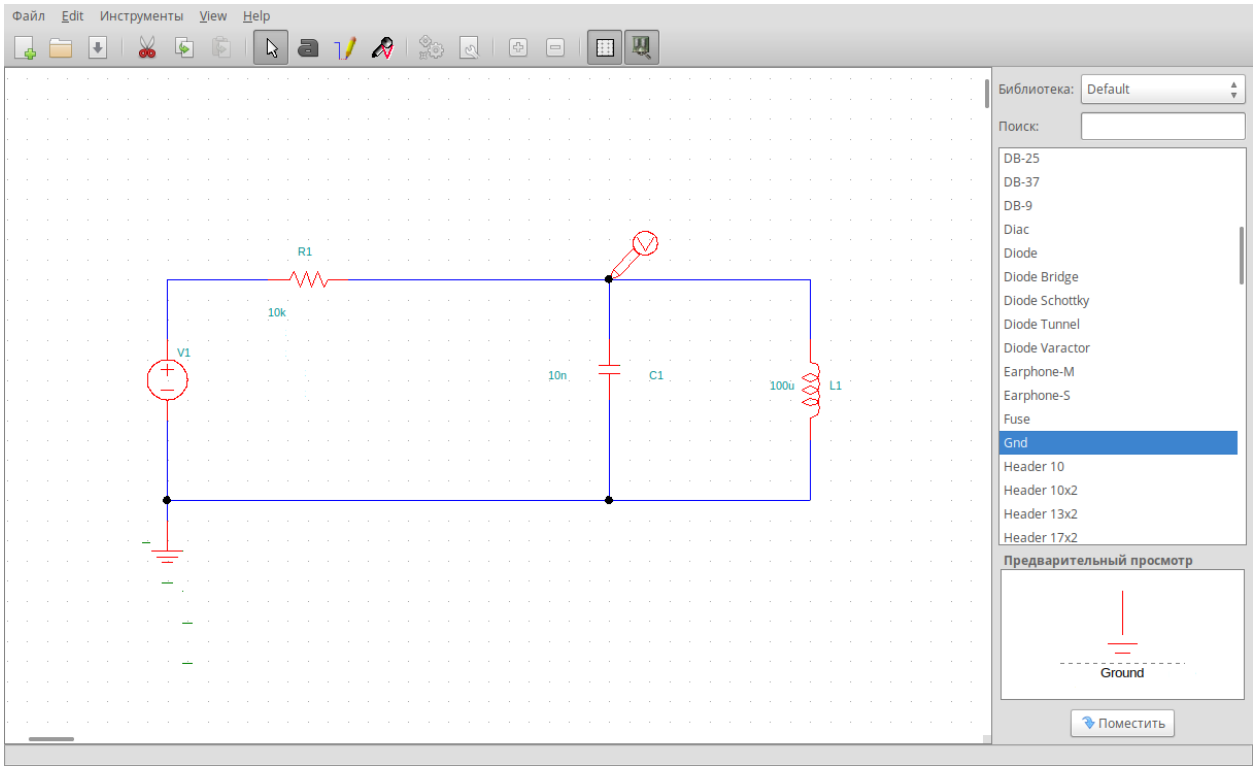


Рис. 15.23. Новая схема – испытание колебательного контура

Программа позволяет увеличивать изображение схемы – достаточно использовать колёсико мышки. В таком масштабе легче различить значения резистора, конденсатора и индуктивности. И, повторив анализ на переменном токе, можно убедиться, что он работает. И, кстати, работает в ngspice. Перед этим я столкнулся с появлением ошибки, связанной с жалобами симулятора, что ему что-то не додали. Собрав схему заново, эту проблему удаётся решить.

Впрочем, и ElementaryOS подчас впадает в глубокую задумчивость, выводя меня из равновесия. Как и поведение мышки в Windows 10 – вращением колёсика мышки не получается перемещать страницу, которая дёргается, но не движется. Видимо, причина одна – стремление к улучшению.

Есть ещё некоторые замечания, если вам захочется поработать с программой.

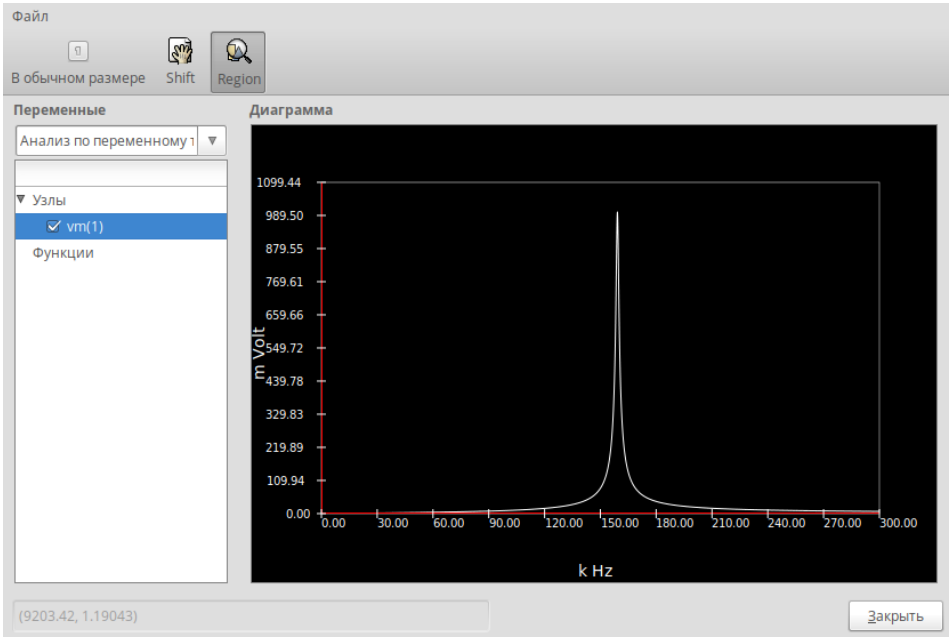


Рис. 15.24. Результат симуляции при построении АЧХ

Первое замечание касается того, что программа, видимо, хранит данные по результатам моделирования в файле со схемой. Если скопировать схему в новый файл, то неприятности с прежним моделированием могут сохраниться. Впрочем, причина этого может быть в другом. Мышкой можно ненароком зацепить и сдвинуть компонент с места. При этом, похоже, связь компонента с цепью прерывается, и её не удаётся восстановить перемещением компонента на место. Проведём эксперименты, чтобы подтвердить или опровергнуть это. Предварительно я хочу сохранить файлы в директории /home/владелец (у меня это /home/vladimir). Причина в нелюбви некоторых программ к кириллице и разделам из двух слов. Схему электрической цепи возьмём простейшую: делитель из двух резисторов и источник напряжения. Итак.

Осуществим моделирование собранной цепи.

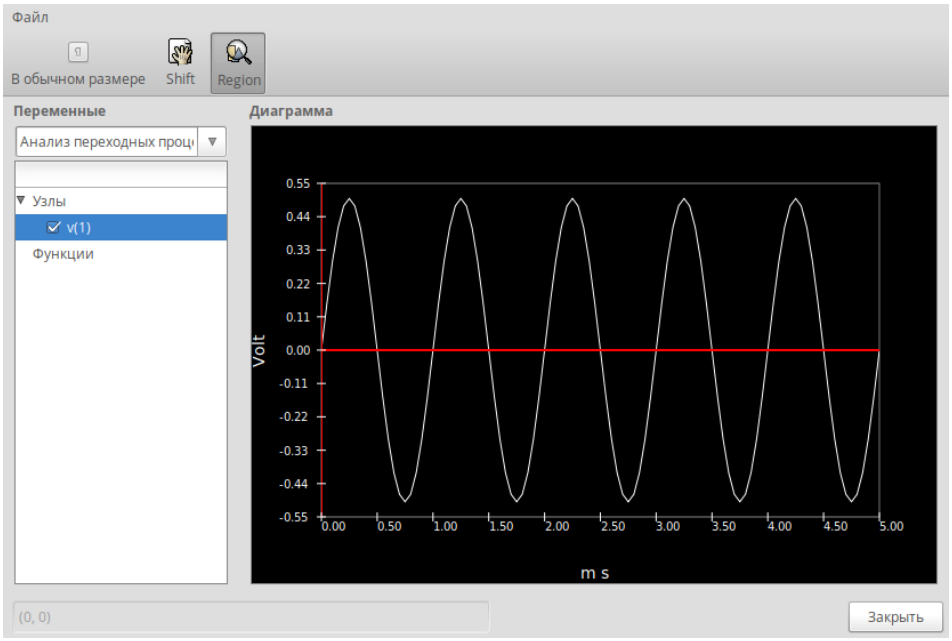


Рис. 15.25. График эксперимента с делителем напряжения

Проведённый эксперимент показал, что предположение было неверным, но само смещение имеет место при неаккуратной работе с мышкой. Обращайте внимание на появление лишних точек в соединении. При необходимости возвращайте компонент на место, в противном случае моделирование будет проходить неверно.

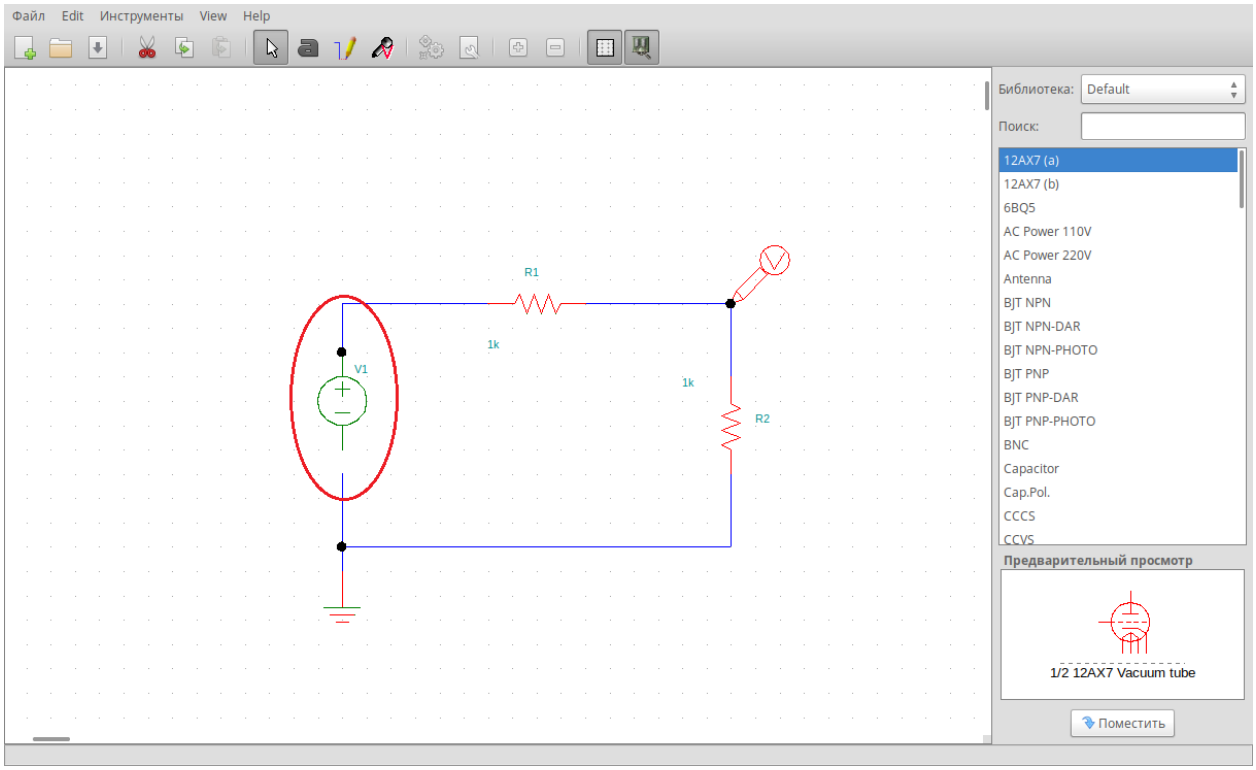


Рис. 15.26. Появление лишней точки соединения при смещении компонента

В этом месте рассказа я снёс установленную версию ElementaryOS по причине её плохой работы в VirtualBox. Новую остановку я произвёл несколько иначе: отказался от установки обновлении при установке ОС, дождался появления на экране установки предложение удалить загрузочный диск и нажать клавишу **Enter**. После установки обновление я предпочёл сделать в терминале:

```
sudo apt-get update
sudo apt-get upgrade
```

ElementaryOS стала работать гораздо лучше, но остался один недостаток – для работы в полноэкранном режиме приходится каждый раз давать терминальную команду:

```
xrandr --output Virtual1 --mode 1680x1050
```

После следующего запуска, увы, разрешение возвращается к минимальному.

В данный момент меня интересовал такой вопрос – влияет ли конкретная модель транзистора на поведение электрической схемы? Дело в том, что АЧХ транзистора не выказывает желание показать верхнюю граничную частоту, что мне не нравится. Вот переходной процесс для идеальной модели транзистора.

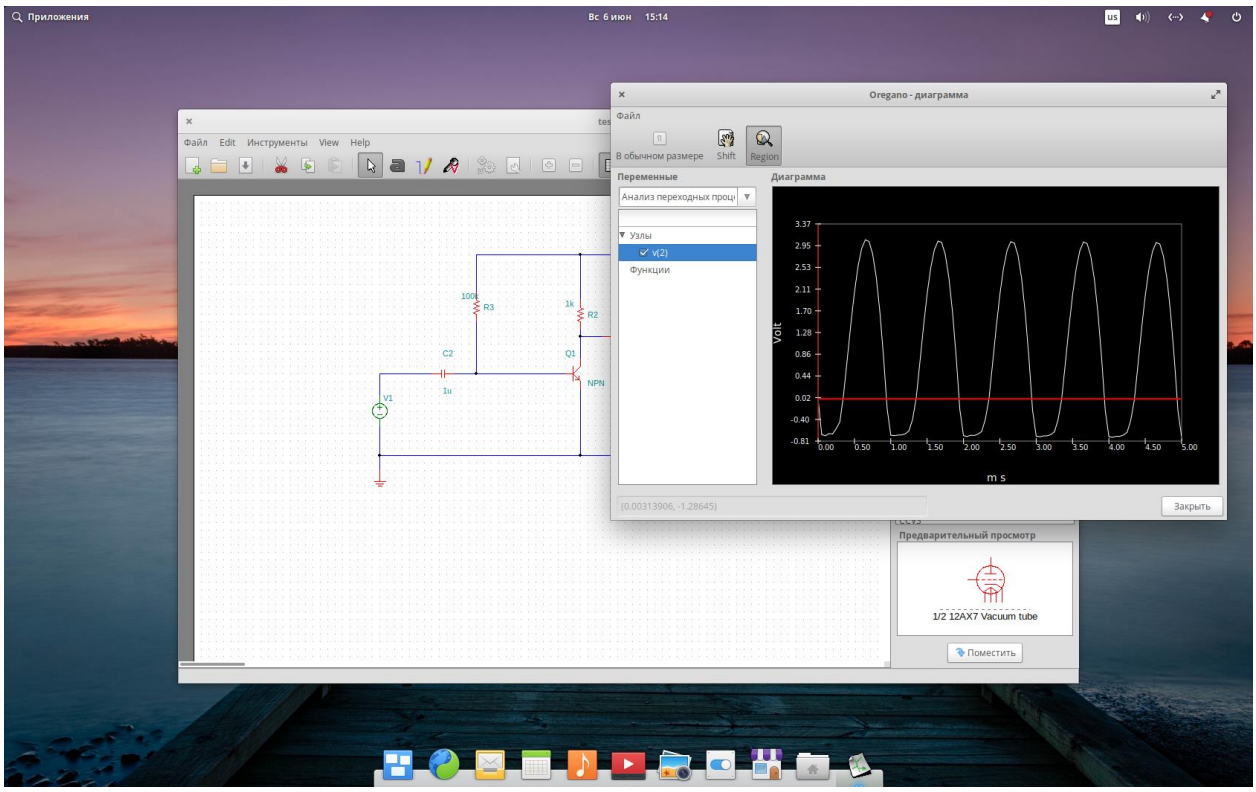


Рис. 15.27. График переходного процесса с NPN транзистором

У меня есть spice-модель транзистора КТ315, которую я добавил в папку моделей, и работа транзистора, признаюсь, изменилась при сохранении прежних настроек.

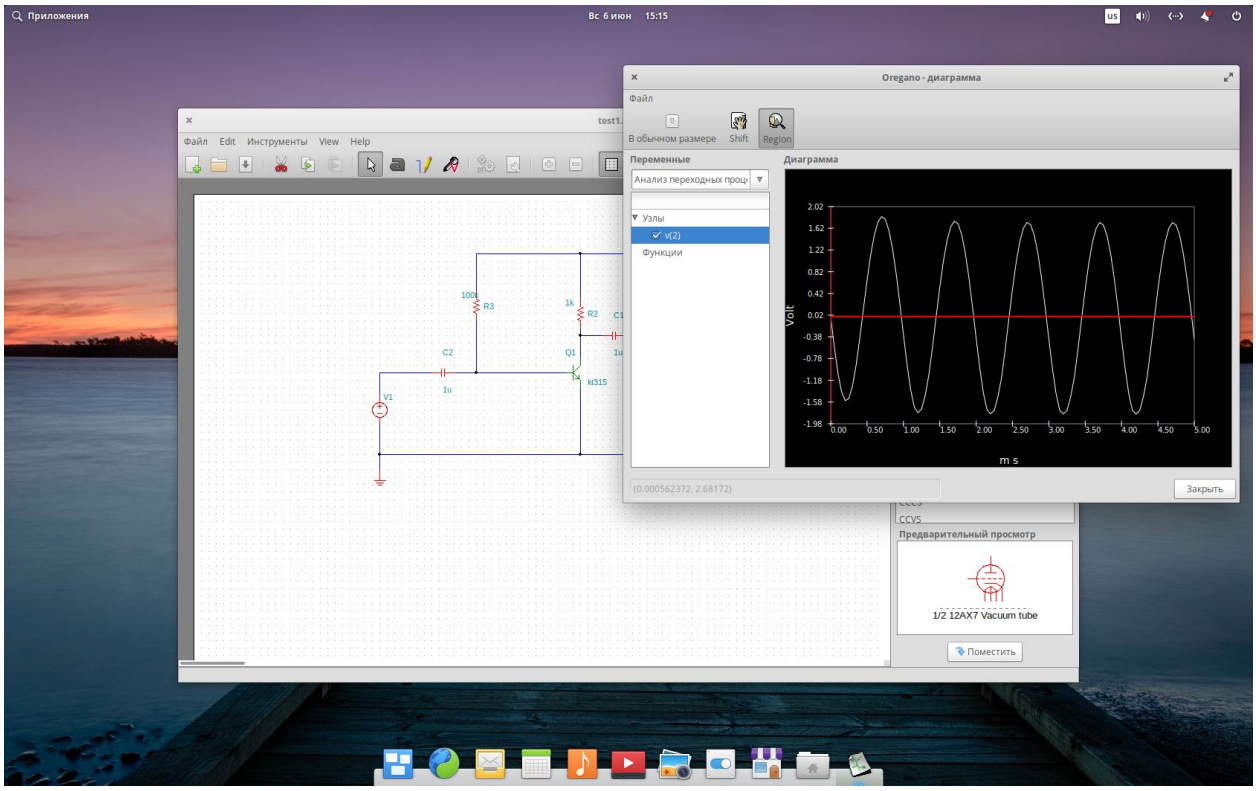


Рис. 15.28. График переходного процесса с транзистором КТ315

Приходится мириться с тем, что частотные свойства транзисторов программа не учитывает. И ещё одно – у меня так и не получилось посмотреть Фурье-анализ. Впрочем, с частотными свойствами транзистора, я этого не исключаю, я поторопился, но не с Фурье-анализом. Я вижу причину этого в версии программы – одна из ранних версий, где было реализовано не всё, что было задумано.

По этой причине я решил удалить программу Oregano, скачать версию для дистрибутива Debian, которая явно более позднего изготовления. К сожалению, я не понял, как установить эту версию, используя штатные функции ElementaryOS, что обусловило мои дальнейшие действия.

Я распаковал пакет (есть штатное предложение в выпадающем меню: распаковать). После распаковки содержимое папки usr я, используя менеджер файлов с правами администратора, перенёс в папку usr дистрибутива. После первого запуска Oregano в терминале понадобилось установить библиотеку, без которой программа не запускалась, попутно установились или обновились ещё библиотеки, что может сказаться впоследствии, но с появлением проблем будем разбираться по мере их возникновения.

На первый взгляд программа изменилась существенно – так папка с моделями пополнилась моделями транзисторов, например, появился транзистор 2N2222. Но и с этим появились проблемы: при использовании GNUCAP появляются сообщения об игнорировании ряда параметров, а ngspice отказывается работать с моделью. Отложим эту проблему в долгий ящик, используем модель транзистора КТ315:

```
.model KT315 NPN (IS=10F BF=584.517 VAF=100 IKF=29.2714M ISE=131.803P
+ NE=2.08337 BR=1.95214 IKR=9.99996M ISC=100.316P RE=1 RC=5.48635
CJE=27.3893P
+ VJE=700.001M MJE=500.287M CJC=27.3893P VJC=700.001M MJC=500.287M
TF=450.287P
+ XTF=499.984M VTF=10 ITF=10.2268M TR=153.383P)
```

С этой моделью работают оба симулятора. Кстати, третий spice3, появившийся в этой версии программы, среди доступных в ElementaryOS не обнаружился.

Меня интересует Фурье-анализ. С GNUCAP анализ не запустился, а с ngspice, если выбрать моделирование переходных процессов и спектральный анализ, работает. Вот сигнал при большом входном напряжении.

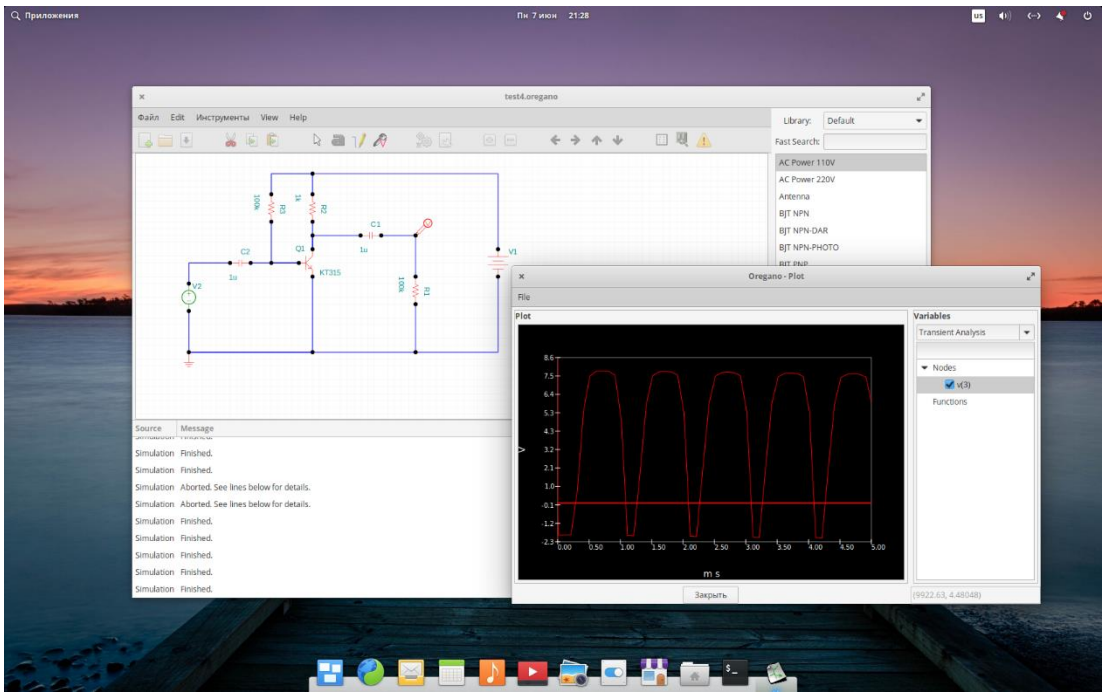


Рис. 15.29. Сигнал на выходе однокаскадного усилителя

И его спектр.

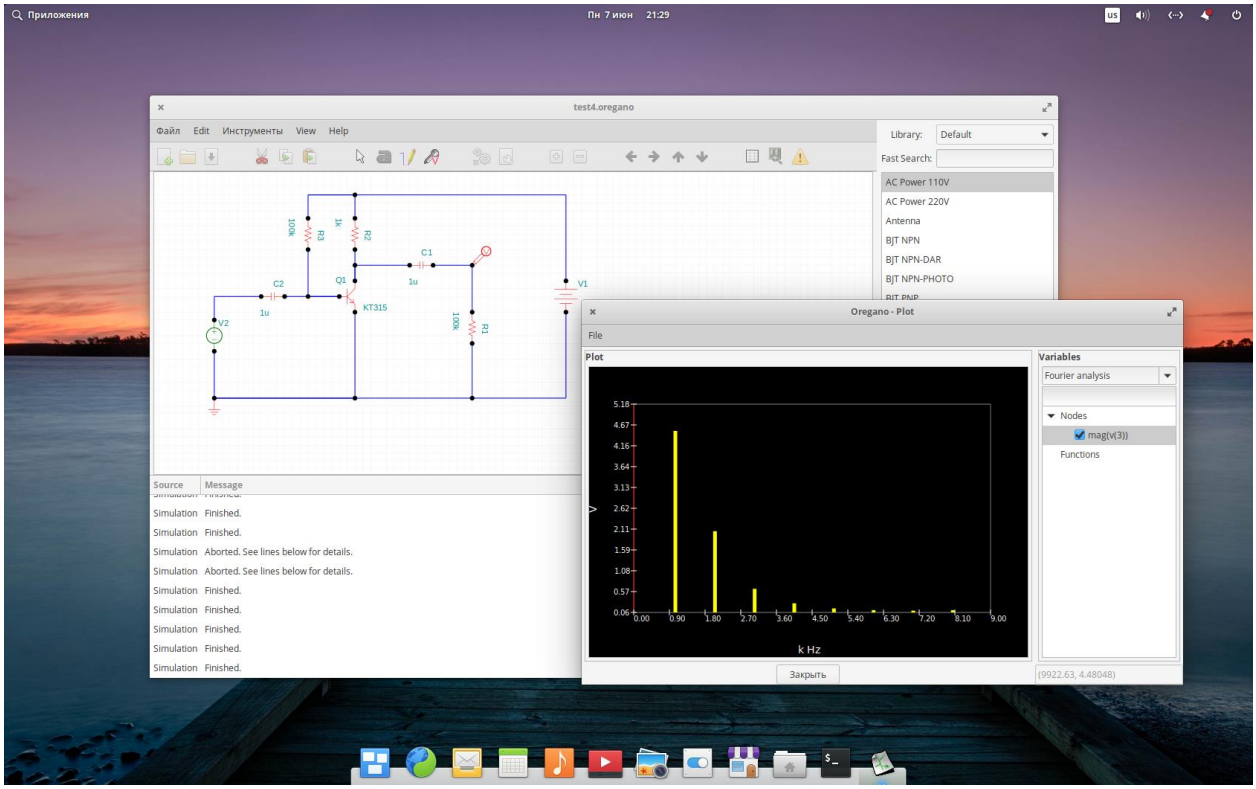


Рис. 15.30. Спектр искажённого синусоидального сигнала

Есть и ещё одна странность – ngspice неверно отрабатывает анализ на переменном токе, не выводя заданный вывод в netlist. Но в GNUCAP можно построить АЧХ, используя децибелы.

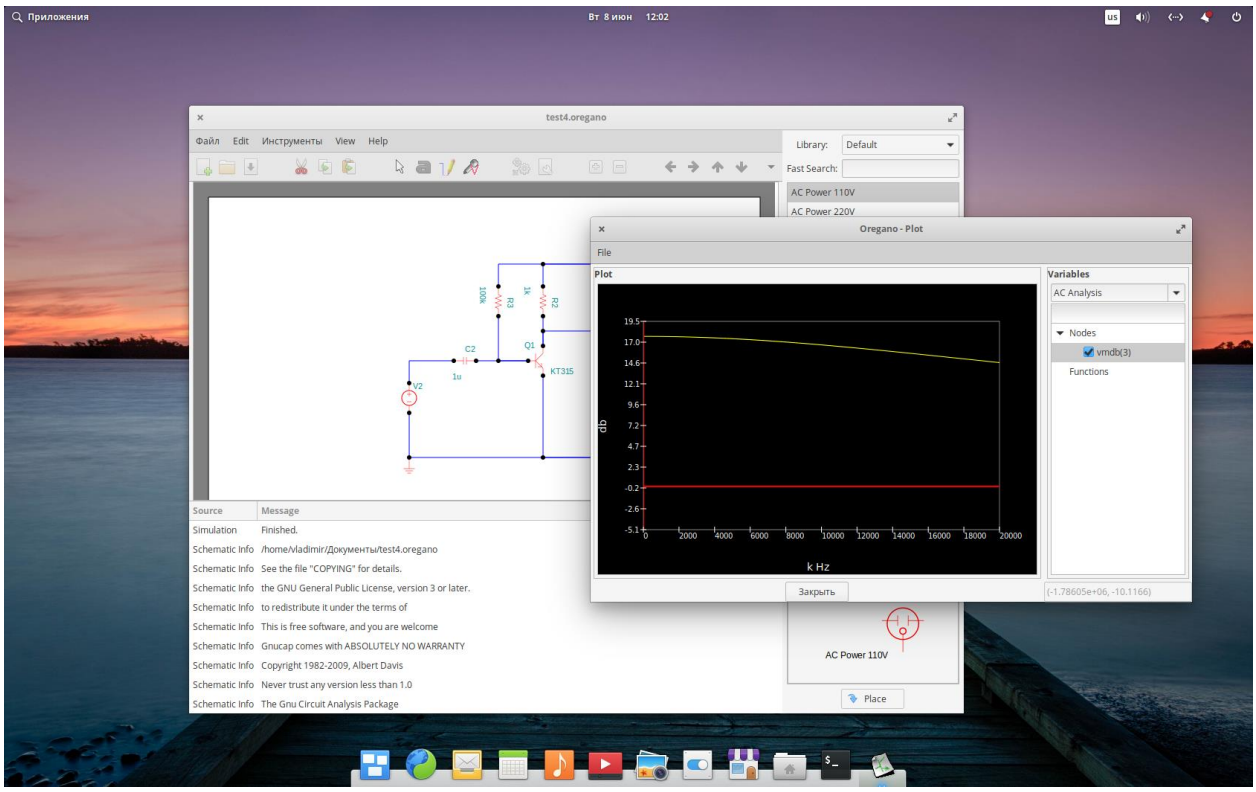


Рис. 15.31. АЧХ с транзистором КТ315

Удовлетворив своё любопытство в части анализа на переменном токе и Фурье-анализа, я хочу попробовать ещё один вид моделирования (или не один). В Oregano есть таймер 555. Стандартная схема генератора…

Увы, не работает генератор. В модели есть описание подсхемы, но я не знаю, где её искать, не знаю, как и можно ли использовать подсхемы в Oregano. Не получилось и попробовать работу D-триггера. Не захотела программа моделировать его, хотя изображение триггера есть. Я не исключаю, что ряд символов служат только для создания рисунков схем. Или это «задел» на будущее. Что ж, можно пользоваться тем, что работает или поискать более современное решение проблемы. Например, попытаться поставить какие-то программы для Debian, но, конечно, в VirtualBox, чтобы не испортить операционную систему непродуманными экспериментами.

Подводя итог, можно сказать, если вам достаточно тех средств, что есть в дистрибутиве – зайти в соцсеть, послушать музыку или посмотреть фильм; если достаточно использовать текстовый процессор и работающую часть программы Oregano, то можно поставить на компьютер дистрибутив ElementaryOS. Во многих отношения симпатичная и удобная операционная система.

Глава 16. Ещё некоторые эксперименты в ElementaryOS

Признаться, я немного расстроился, не обнаружив среди доступных программ знакомую мне программу Qucs. С ней я познакомился давно…

Qucs (2006-2007 г.)

Завершить же эту экскурсию в прошлое (в книге «Наглядная электроника») я хочу одним мелким, но для меня показательным экспериментом. Еще со студенческих времен, если я ничего не путаю, я запомнил, что в трехфазной сети при одинаковой нагрузке на фазы ток через нулевой провод не протекает. Я помню векторную диаграмму, которая это хорошо доказывает, но мне ни разу не приходилось измерять этого. Не довелось.

Интересно, что мне покажет какая-нибудь из программ в этом эксперименте.

Для начала запустим в Linux программу Qucs:

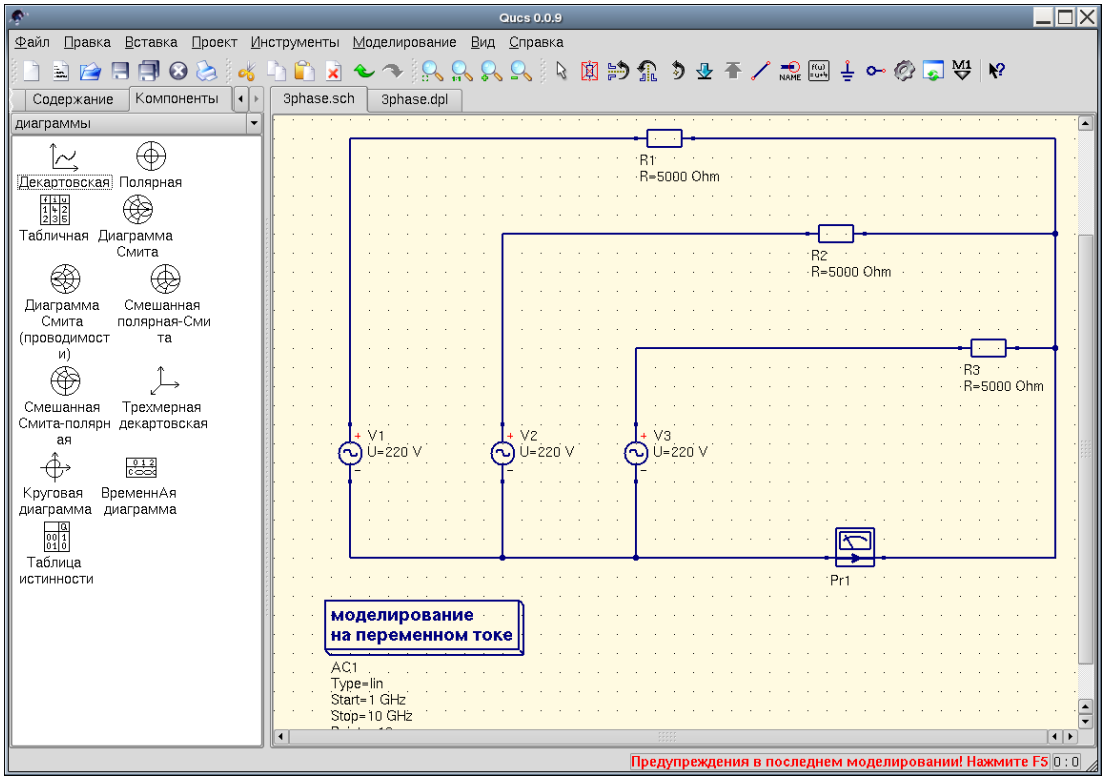


Рис.16.1. Измерение в трехфазной сети при равных нагрузках

А теперь посмотрим, что покажет моделирование схемы:

acfrequency	Pr1.i
1e9	6.94e-18-j6.94e-18
1.5e9	6.94e-18-j6.94e-18
2e9	6.94e-18-j6.94e-18
2.5e9	6.94e-18-j6.94e-18
3e9	6.94e-18-j6.94e-18
3.5e9	6.94e-18-j6.94e-18
4e9	6.94e-18-j6.94e-18
4.5e9	6.94e-18-j6.94e-18
5e9	6.94e-18-j6.94e-18
5.5e9	6.94e-18-j6.94e-18

Рис.16.2. Результаты измерения в трехфазной сети

Как мне и запомнилось, ток в общем (нулевом) проводе, если отбросить комплексный характер отображения результатов измерения, если перевести на обычный язык, составляет столь малую величину, что реально включенный в нулевой провод амперметр переменного тока, скорее всего не отклонился бы от нуля, поскольку модель показывает значение с показателем степени -18. Калькулятор при попытке вычислить это значение показывает откровенный нуль.

Qucs и ElementaryOS

Программа есть в дистрибутиве Debian, но попытка установить программу в ElementaryOS завершилась неудачей. И на этом можно было бы поставить крест, если бы не одно обстоятельство – сегодня существует значительно обновлённая версия QucsStudio. Программа создана пока только для Windows. Но меня интересует, будет ли программа работать в Linux с подержкой Wine. Последнее приложение нужно установить, я использую установку в терминале. Добавив необходимую библиотеку (при запуске программы через терминал можно увидеть, какую библиотеку следует добавить), я копирую QucsStudio из Windows в папку /home/vladimir. Запускать приходится тоже терминальной командой:

```
wine /home/vladimir/qucsstudio/bin/qucs.exe
```

И программа запускается. Опробуем простое моделирование.

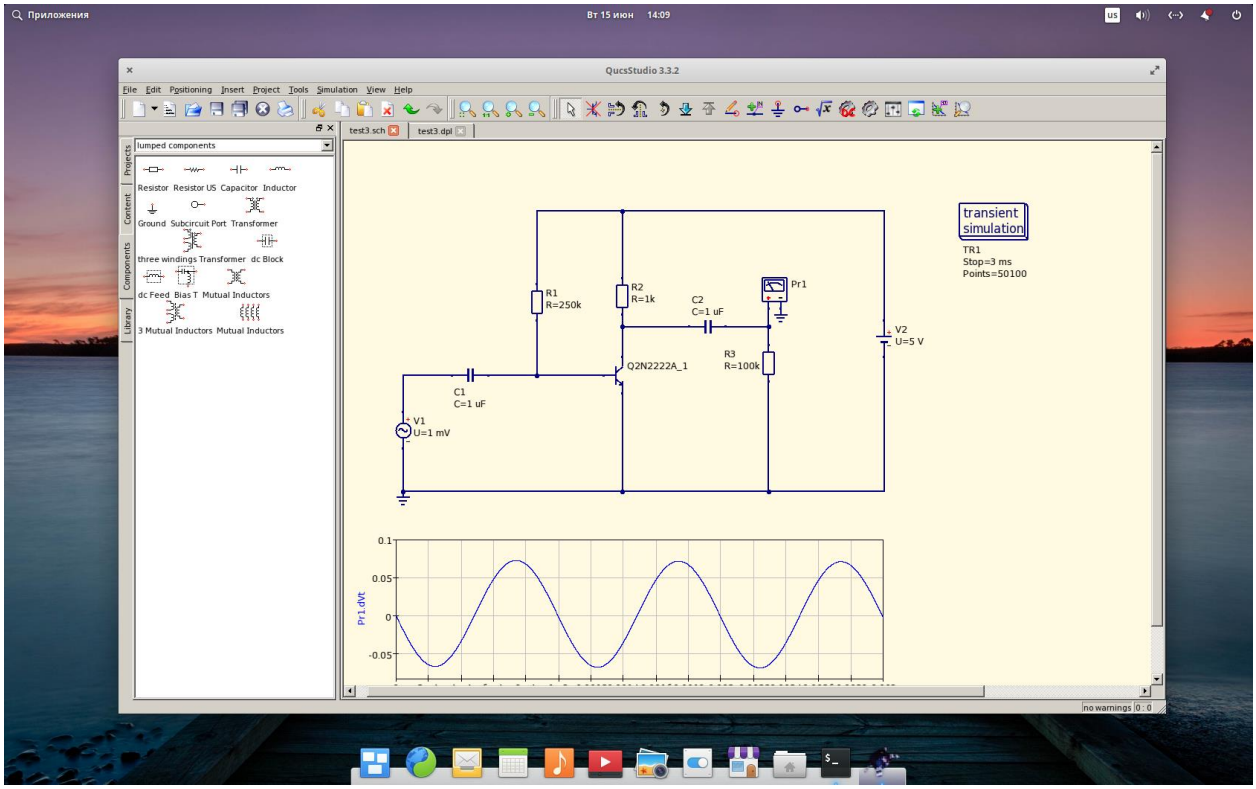


Рис. 16.3. Моделирование переходного процесса схемы усилителя

Хорошо бы проверить все режимы моделирования и разные схемы, но на это нужно время – каждую схему нужно собрать, чтобы опробовать, исправить при необходимости ошибки (и это случается). Например, в программе есть режим показа постоянных напряжений, что позволяет определить начальные напряжения: выбирая сопротивления, следует проверять напряжение на коллекторе транзистора. Проверив напряжение на коллекторе транзистора 2N2222A, я решил повторить эту операцию с другим транзистором из библиотеки (закладка Library слева). И вот результат, который меня, признаюсь, очень смутил.

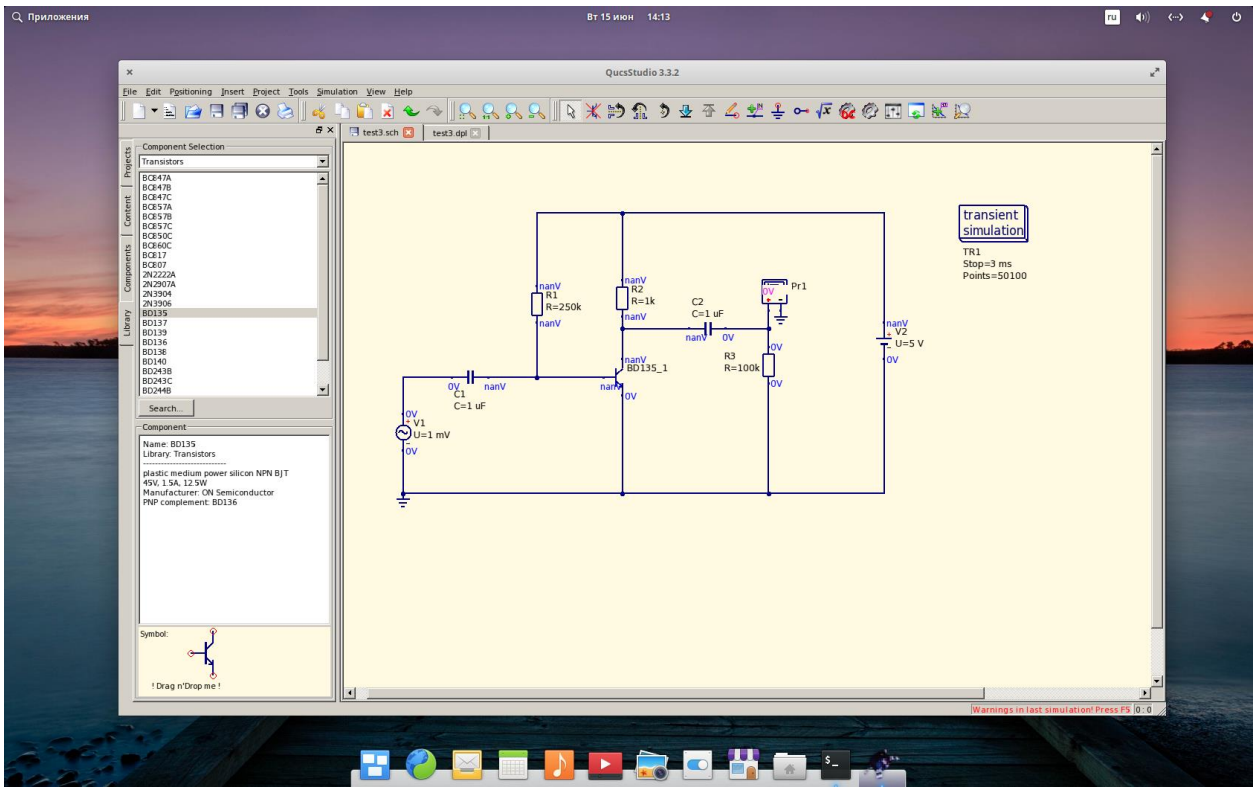


Рис. 16.4. Результат проверки режима начальных напряжений с транзистором BD135

Однако, повторив ту же проверку в Windows, я убедился, что результат тот же. Это не проблема работы программы в wine. И построение АЧХ в QucsStudio проходит вполне удачно.

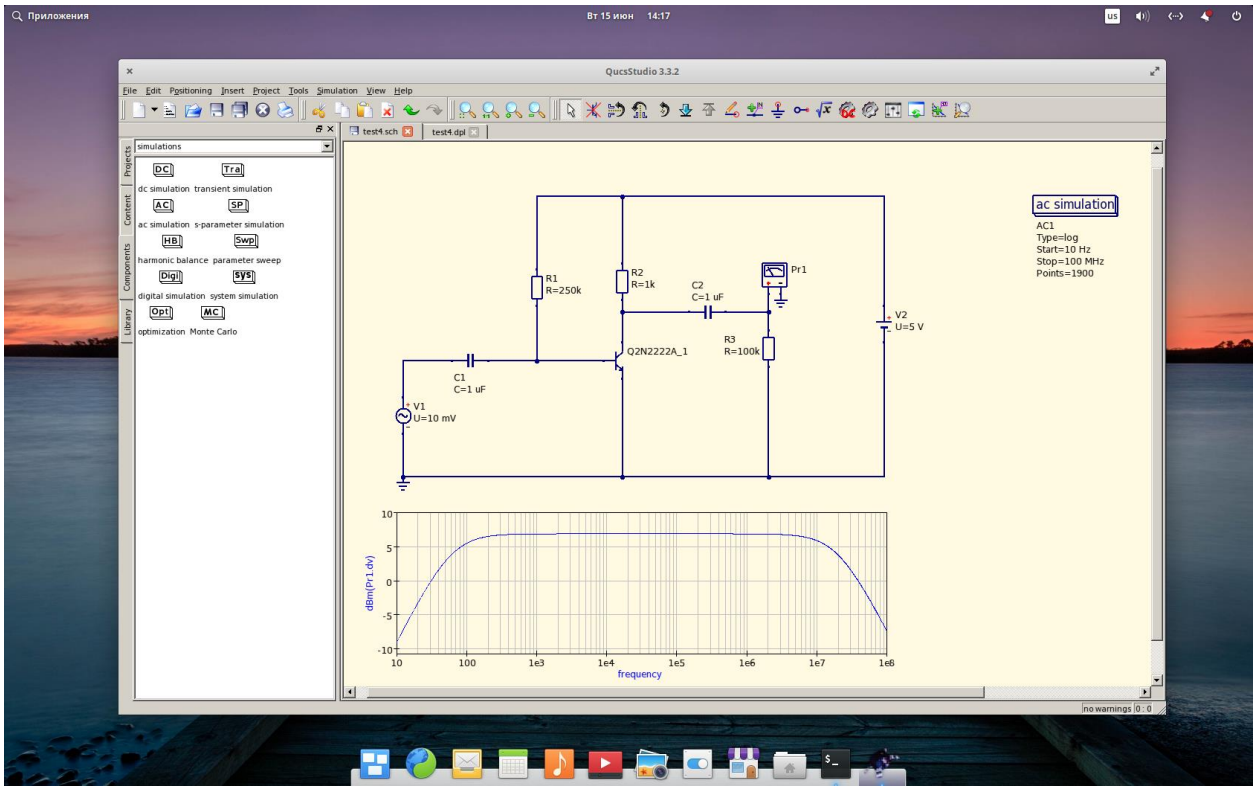


Рис. 16.5. Построение АЧХ в децибелах (относительно входного напряжения 1 мВ)

На этом графике удобно определять частоты среза, проверить равномерность частотной характеристики, словом, удобно. А последнее, что я хочу проверить, не терпится мне опробовать ещё одну программу, проверить простой делитель частоты на D-триггере.

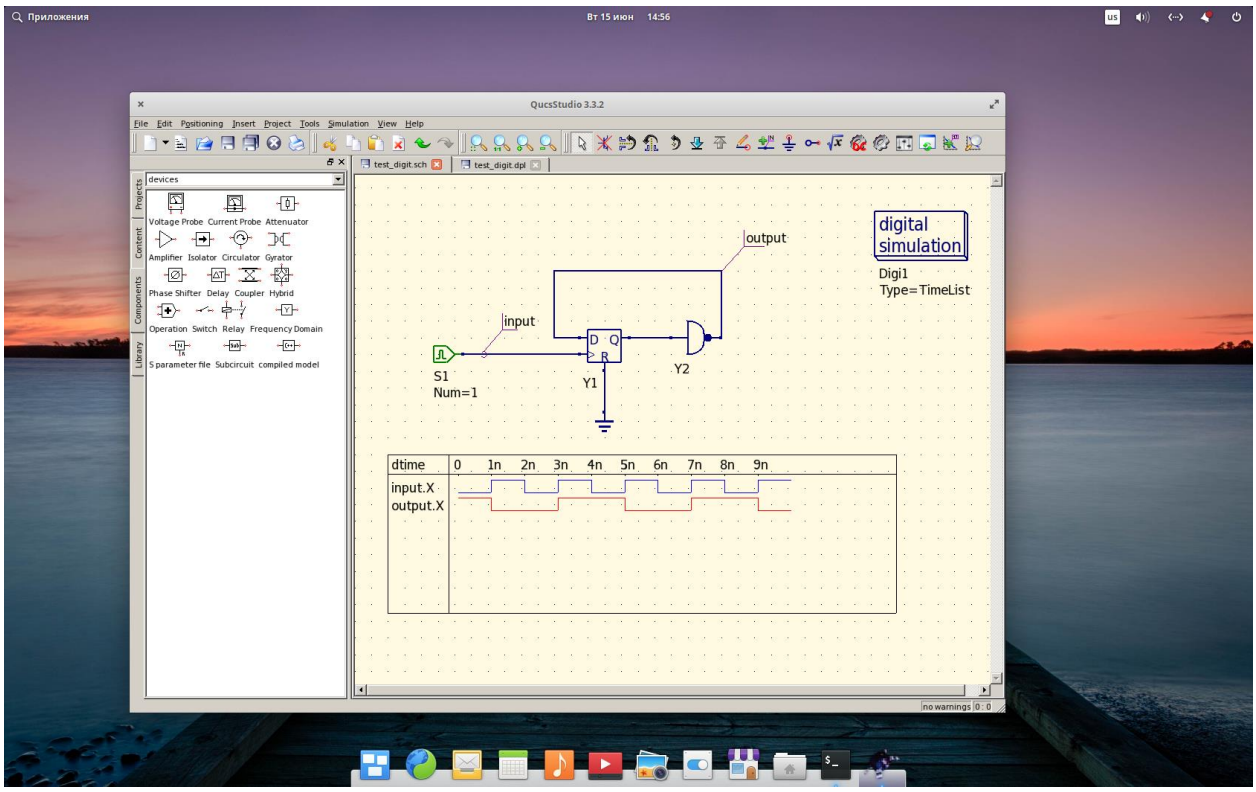


Рис. 16.6. Проверка делителя частоты на D-триггере

О том, как работать с программой Qucs и QucsStudio я рассказывал много раз. Не буду сейчас повторяться. Итак, куда я так спешу?

Есть весьма полезная для радиолюбителей программа SimulIDE. Меня вдохновил пример работы QucsStudio в среде wine, я решил повторить это с программой SimulIDE. И ничего у меня не получилось – требовалась поддержка Qt5 для работы программы, а штатная поддержка в wine только для предыдущей версии. Я бы не стал упоминать об этом, если бы не существовала версия SimulIDE для 64-битового дистрибутива Linux. Загрузив последнюю версию (тестовую), другие версии загрузить не получилось, распаковав её, я запустил программу (в домашней папке или директории, если угодно) из терминала. Как и положено не хватило библиотек для успешной работы.

При запуске программы в терминале появляется сообщение об отсутствии нужной для работы программы библиотеки. При запуске программы из проводника нет такой информации. Вот почему многие предпочитают работать с терминалом.

SimulIDE и ElementaryOS

Нужные библиотеки не обозначились все сразу, они появлялись по одной. Для поиска каждой из них можно в терминале использовать команду:

`apt-cache search libqt5…` Далее следует нужная библиотека, например, libqt5serialport5.

Установка этих библиотек, напомню, командой:

```
sudo apt-get install libqt5serialport5
```

Обратите внимание на небольшое отличие написания библиотеки в дистрибутиве и том, о котором говорит программа SimulIDE, когда жалуется на нехватку библиотек. Различие небольшое.

Добавив все необходимы библиотеки, можно запустить программу из терминала командой:

/home/vladimir/SimulIDE/bin/simulide

У вас папка пользователя, конечно, не vladimir.

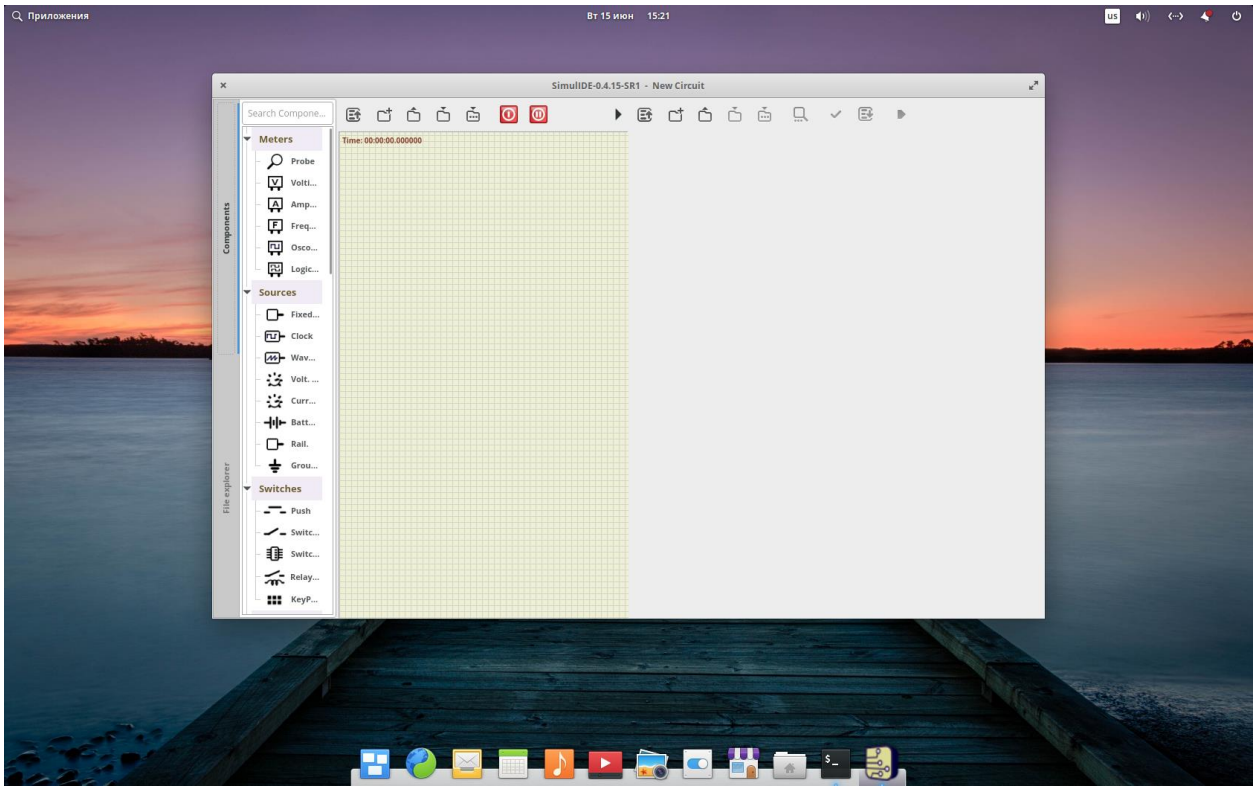


Рис. 16.7. Запуск программы SimulIDE в ElementaryOS

Хотя это тестовая программа, признаюсь, я не пожалел, что установил её. И вот почему. С последней встречи с этой программой прошло время, программа обновилась, и появился, например, такой компонент, как осциллограф в «новом одеянии».

И пока не забыл, хотелось бы отметить, что среди цифровых микросхем вы можете увидеть отечественные, аналогов которым, похоже, нет.

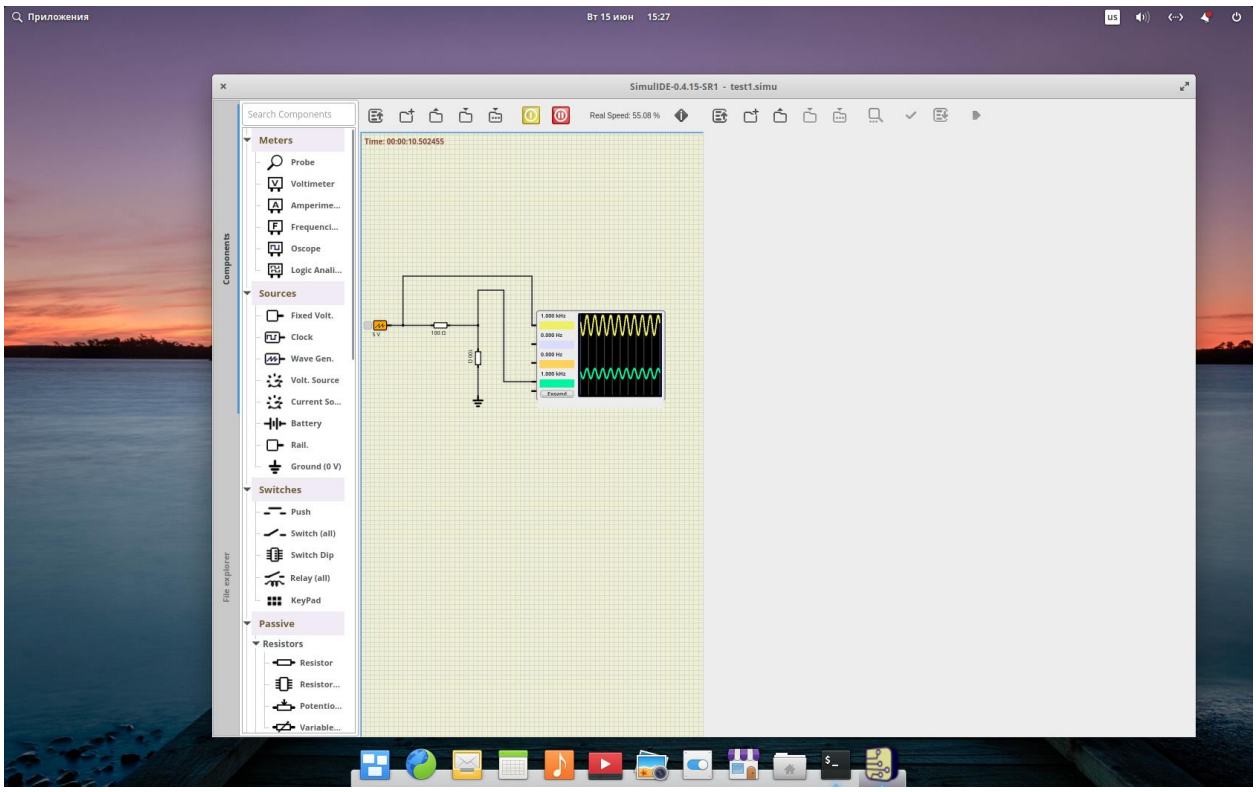


Рис. 16.8. Работа с новым видом осциллографа

В нижней части панели есть кнопка **Expand (развернуть)**, нажав на которую вы получите развёрнутый вид панели осциллографа, с которым удобнее будет работать.

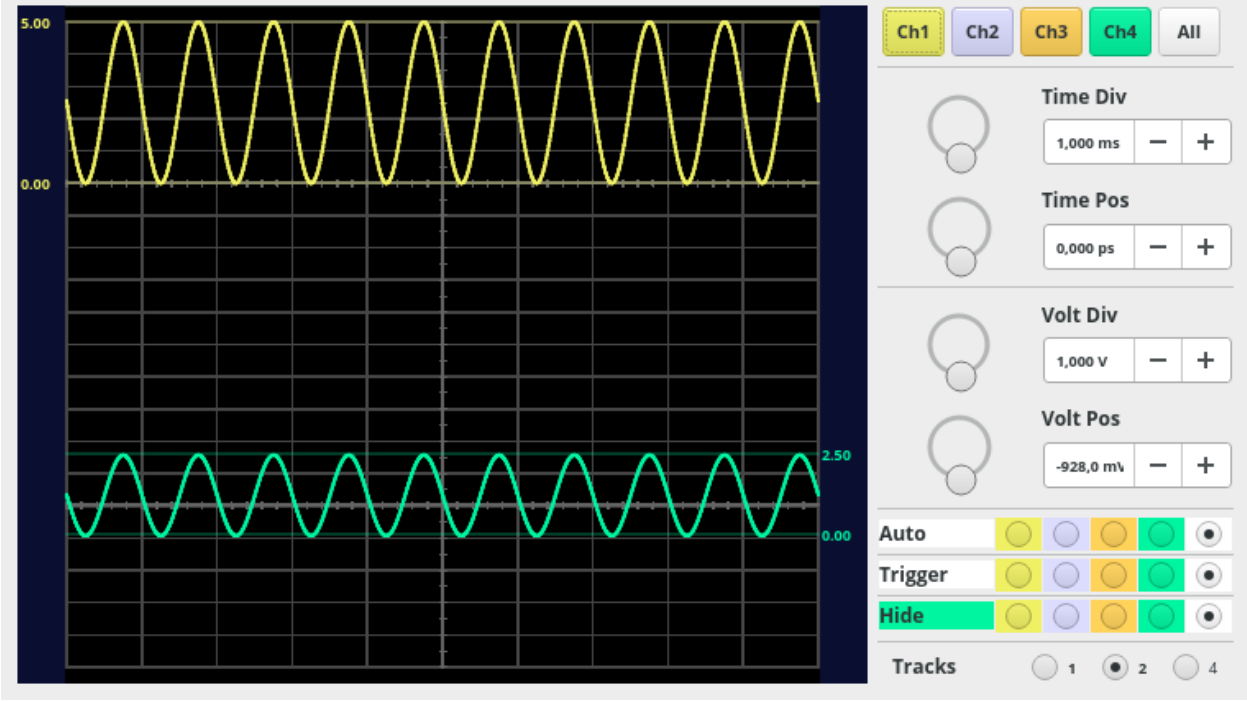


Рис. 16.9. Развёрнутый вид экрана и панели осциллографа

Я надеюсь, что программа работает одинаково и в Linux, и в Windows. Поэтому продолжу знакомство с новой версией в Windows. Но прежде я хочу иметь в ElementaryOS ещё одну программу (скорее, среду разработки) под названием Arduino. Отыскать её не сложно с помощью

поиска в терминале, и устанавливается она аналогично другим программам. При установке устанавливаются (я надеюсь) все необходимые библиотеки. А при первом запуске появляется такое сообщение:

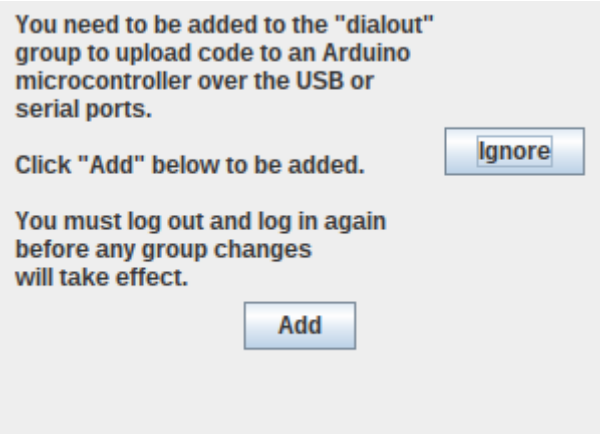


Рис. 16.10. Первый запуск программы проекта Arduino

О чём сообщает программа? О том, что для загрузки программы через USB или COM-порт вам следует быть в группе dialout. Чтобы добавить себя в эту группу, нажмите кнопку **Add**. Мне нравится такая забота о пользователе – раньше нужно было самому отыскивать решение этой проблемы. После выхода из графического режима (чтобы не перезагружаться полностью) и повторного входа можно полностью работать с программой.

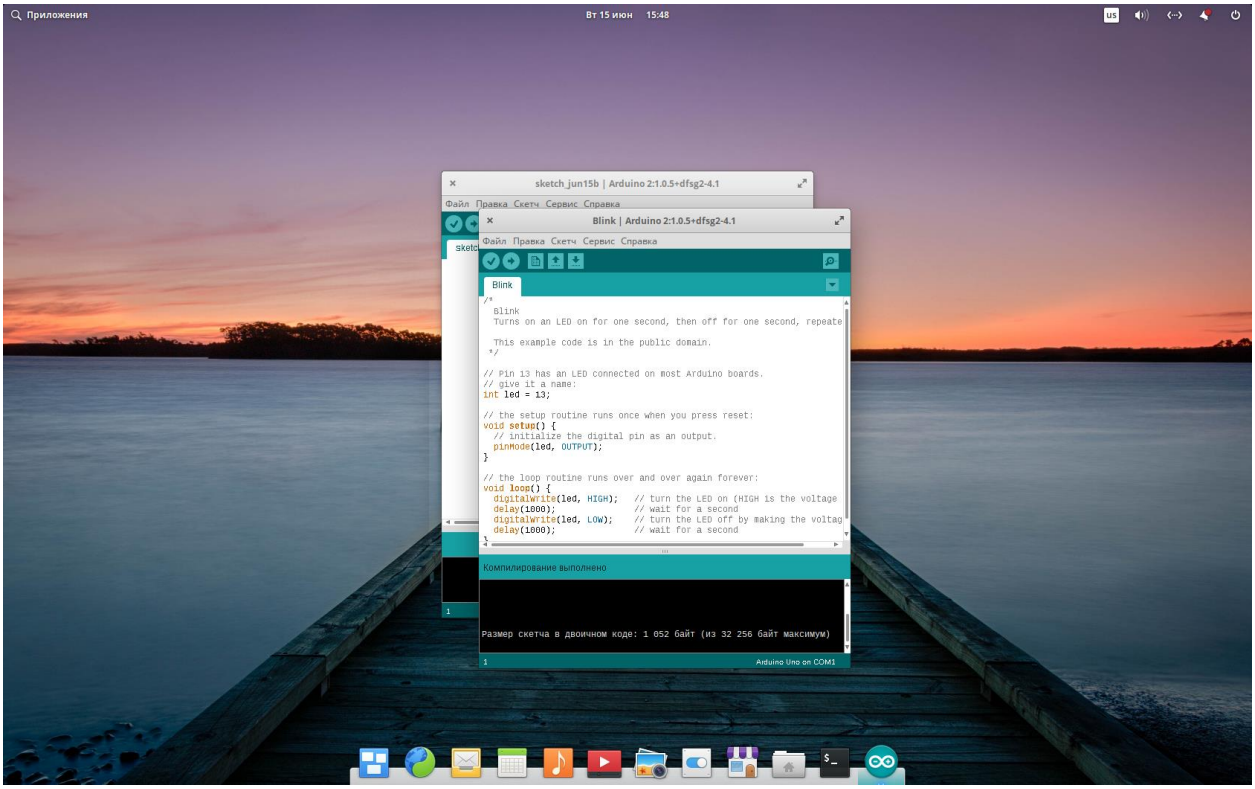


Рис. 16.11. Запуск программы Arduino с примером из набора готовых решений

Отчего я вспомнил про Arduino? В первую очередь, конечно, по причине прекрасной среды для освоения работы с микроконтроллером. А потом – программа SimulIDE позволяет проверить работу ряда микроконтроллеров (и модуля Arduino) с внешними элементами электрической схемы.

Сказав, что программа одинаково работает и в Linux, и в Windows, я несколько погорячился. Более точно – то, что я хотел показать, одинаково (надеюсь, пока) не работает в этих операционных системах. Но оба случая позволяют осуществить обходной манёвр.

О чём идёт речь?

Создаём новый проект в SimulIDE. На панели компонентов находим Arduino Uno и добавляем в окно редактора.

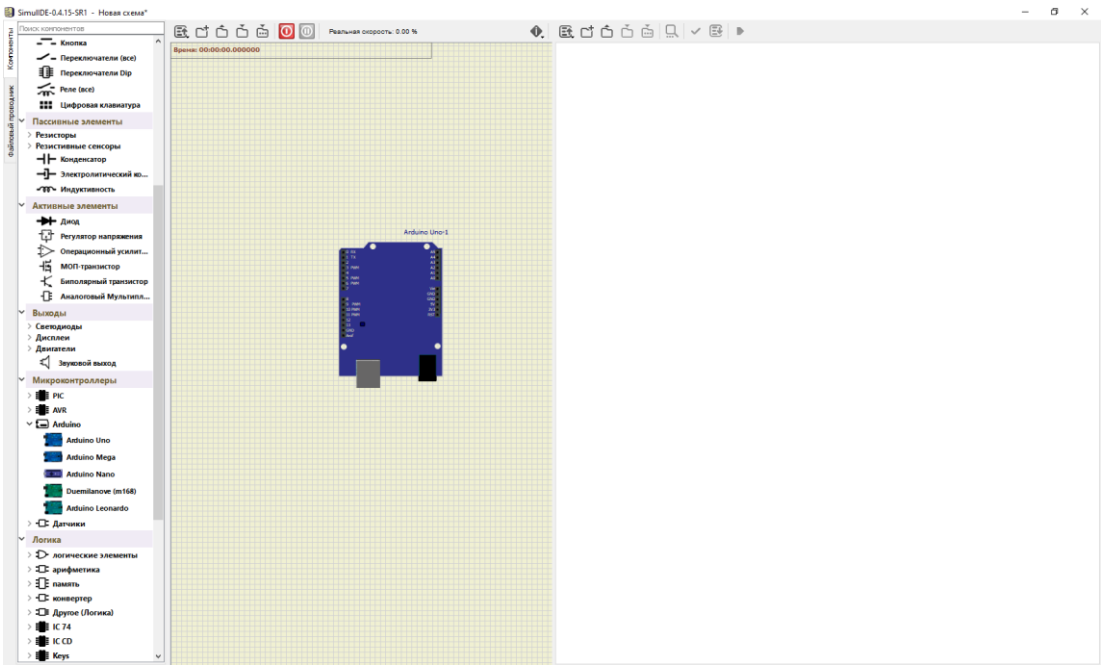


Рис. 16.12. Начало работы с Arduino

С помощью колёсика мышки можно увеличить размер объекта, что будет удобнее при проведении соединений. Добавим резистор и светодиод. В итоге получим такую конструкцию.

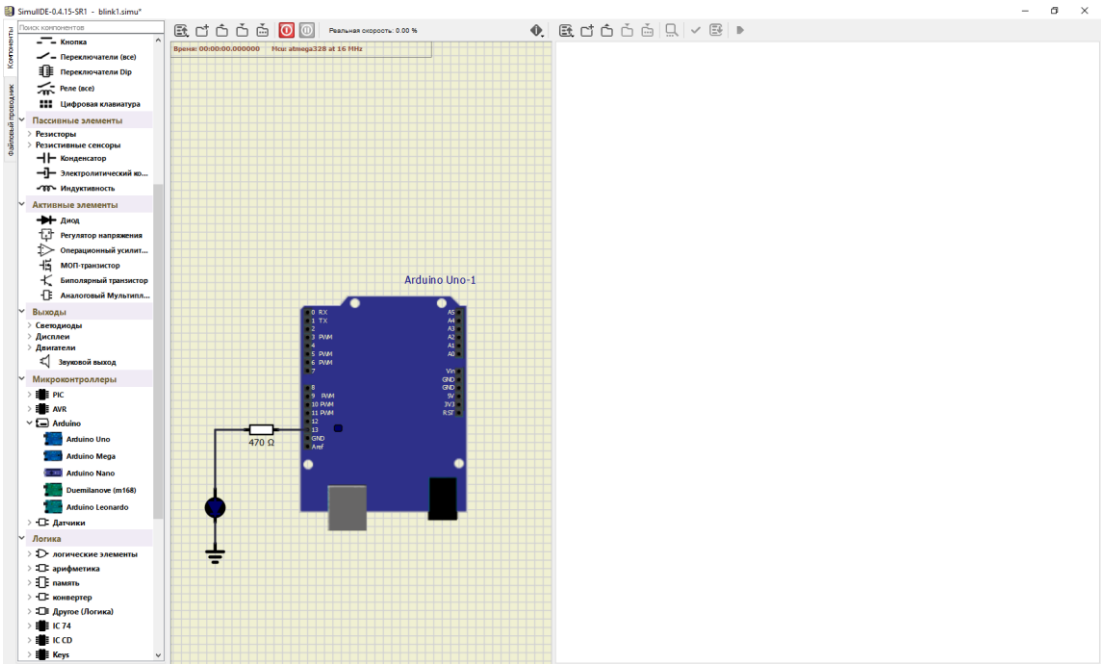


Рис. 16.13. Сборка нужной схемы

Изменить значение резистора несложно – это доступно в свойствах объекта. В Windows достаточно вызвать окно свойств с помощью двойного щелчка левой клавишей мышки по объекту. В Linux у меня так не получается, но щелчок правой клавишей мышки вызывает выпадающее меню, где есть раздел свойств.

Теперь обратимся к правой части окна SimulIDE. Создадим новый файл для записи программы на языке Arduino. Пусть будет простейшая программа мигания светодиодом (в примерах есть более сложные программы). Запишем её.

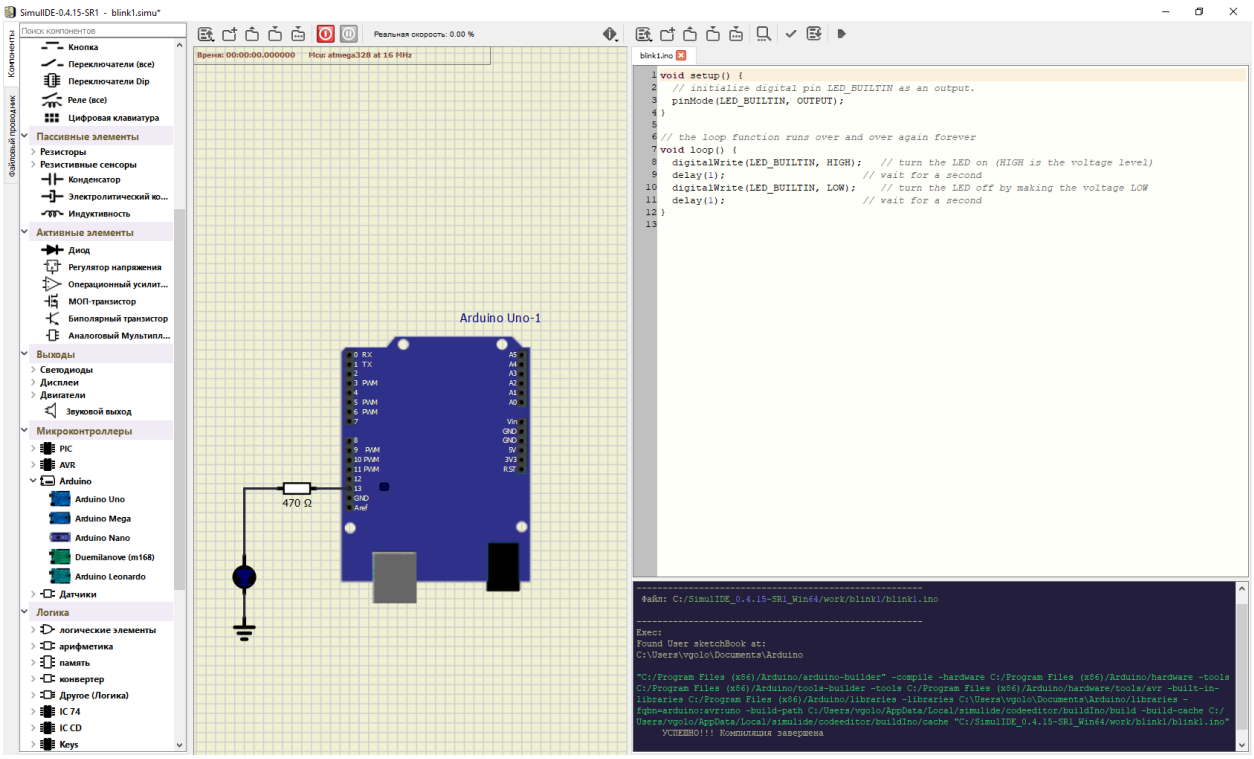


Рис. 16.14. Запись программы для Arduino

При первой попытке компиляции программы (иконка с «галочкой») вас попросят указать место, где находится программа Arduino. Это выполнить не так сложно. Но в Windows приходит сообщение об успешной компиляции (правое нижнее окно сообщений), но найти нужный hex-файл не получается ни вручную, ни с помощью механизма загрузки файла в модель Arduino (иконка, под которой появляется подпись «Загрузить»).

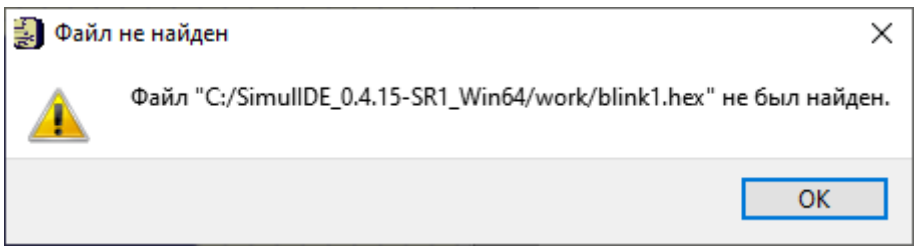


Рис. 16.15. Неудачная попытка загрузить hex-файл в модель Arduino

Обходной вариант – это компилировать программу с помощью программы Arduino, где есть возможность получить нужное, если воспользоваться командой из основного меню Скетч -> Проверить/Компилировать. И, кстати, попытка полученный файл поместить по адресу, указанному выше, не даёт желаемого результата. Для упрощения оставшейся части процедуры я создаю на

диске C: папку work, куда и помещаю hex-файл. Теперь можно открыть свойства модуля Arduino, чтобы указать путь к файлу.

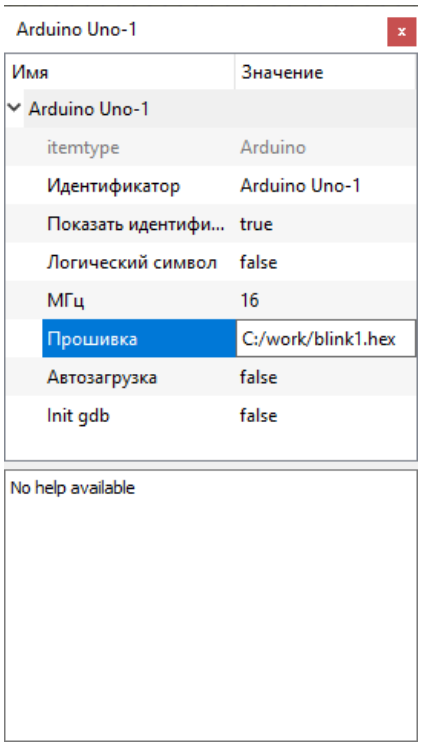


Рис. 16.16. Путь к рабочему файлу для Arduino

Наконец, я добавлю логический анализатор, который очень полезен в ряде случаев при работе с микроконтроллерами. Запустив моделирование, можно увидеть результат.

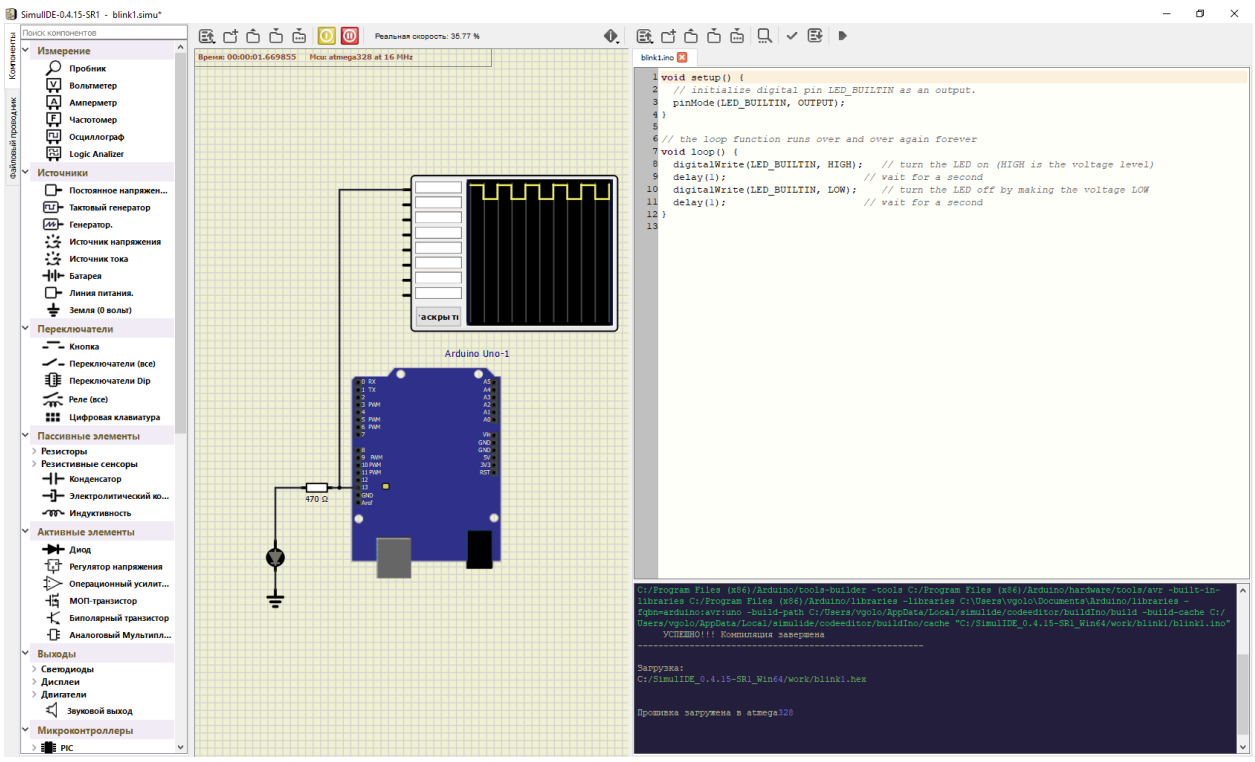


Рис. 16.17. Результат работы программы для Arduino

Как и осциллограф, логический анализатор позволяет изменить вид экрана с помощью кнопки «Раскрыть».

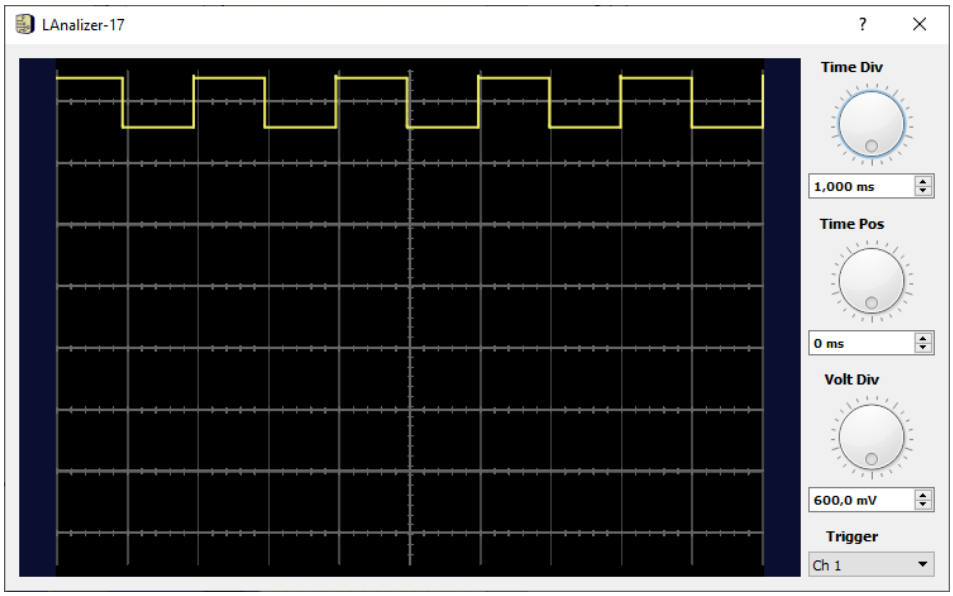


Рис. 16.18. Окно логического анализатора

Четыре канала осциллографа – это хорошо. Но иногда требуется больше каналов для отображения сигналов. От старых историй про микроконтроллеры у меня осталось несколько примеров для PIC16F628A. Один из них я хочу посмотреть в программе SimulIDE. Начнём с добавления на страницу нового проекта контроллера, который выбираем из набора pic-моделей. Добавим логический анализатор из коллекции приборов.

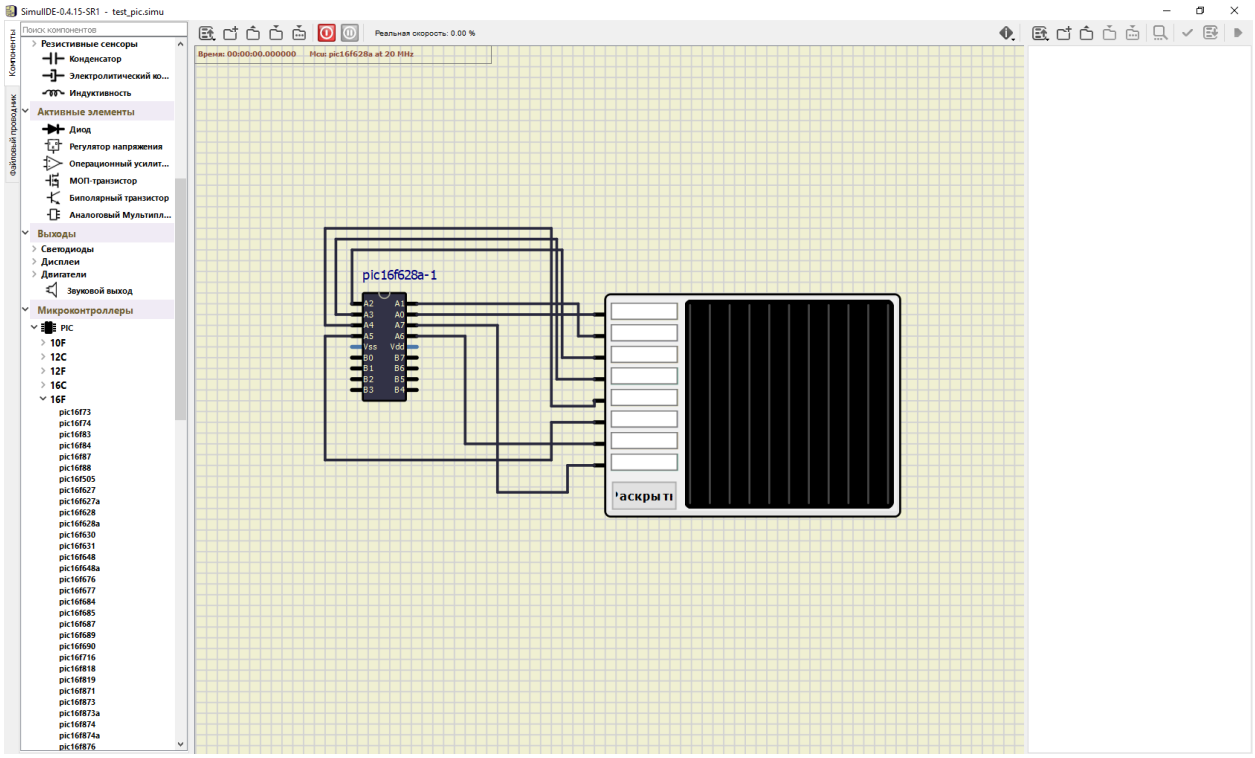


Рис. 16.19. Схема проверки работы PIC16F628A

Для загрузки программы в микроконтроллер воспользуемся выпадающим меню. Щелчком правой кнопки мышки вызываем меню, где ищем команду «Загрузить прошивку», указываем место, где лежит hex-файл, подтверждаем кнопкой «Открыть» проводника. Можно запустить моделирование.

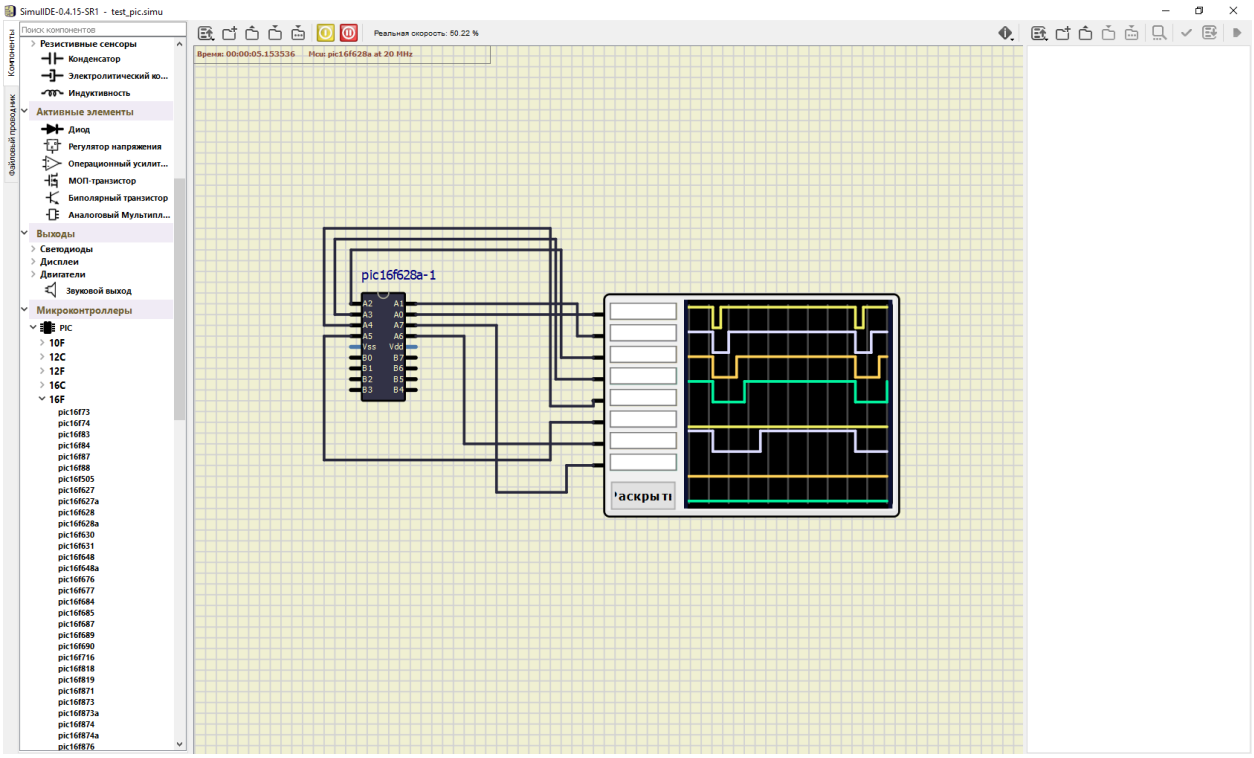


Рис. 16.20. Моделирование работы программы для PIC16F628A

Хотелось бы, несомненно, иметь ещё удобную среду разработки для PIC-контроллеров, но…

Есть хорошая, профессиональная среда разработки MPLAB X. Её можно скачать для Linux с официального сайта. После скачивания и распаковки вы получаете файл MPLABX.sh. Его удобно разместить в домашней папке, а в терминале использовать команду:

`./MPLABX.sh` (может потребоваться ввод команды с добавлением sudo).

Этим начинается установка среды разработки для контроллеров PIC. Что я и пытался осуществить, но столкнулся с нехваткой места на виртуальном жёстком диске (отвёл изначально 16 Гб под операционную систему). Удалив некоторые программы, которые установил по ходу этого рассказа, я попытался повторить установку (программа довольно тяжёлая, установочный файл весит около 500 Мбайт). Закончилось это тем, что виртуальная ОС перестала меня пускать к себе – запуск застревает на вводе пароля при входе. Пароль вводится, принимается, но затем опять появляется окно ввода пароля. Что я понимаю в качестве тонкого намёка мне – пора завершать эту историю.

Пора, значит пора. То, что любителям электроники вполне можно использовать ElementaryOS, я, надеюсь, показал. Остальное зависит только от них.

Глава 17. В преддверии выхода Windows 11 (лето 2021 г.)

Будет ли эта версия лучше предыдущей? Посмотрим, если получится обновить Win10 до Win11. Почему у меня возникли сомнения? Сейчас можно установить Linux внутри Windows, используя wsl. Но у меня работает только wsl1 – мой процессор не поддерживает wsl2. Тем не менее, любопытство не порок, но заставляет делать много лишнего.

Я решил поставить сервер для работы с «x-сами», VcXsrv, и поставить доступные дистрибутивы. Установка, как любая установка полновесной ОС, длится довольно долго, поэтому я не перебрал все доступные дистрибутивы (из тех, что предлагает Microsoft). Самым полным, похоже, получается Debian. «Танцы с бубном» приходится выполнять для совместной работы сервера и дистрибутива. Так запуск Linux после запуска сервера (режим одного большого окна), нужно выполнять из терминала (он появляется после запуска Debian) командой:

```
sudo service dbus restart > /dev/null; DISPLAY=:0 XDG_SESSION_TYPE=x11 gnome-session > /dev/null > /dev/null
```

А разочаровало меня то, что из дистрибутива я не смог работать с сетью – обновление и установка только из терминала. Не очень удобно. Словом, не получилось хорошо и весело. И я не исключаю, что с Windows 11 получится так же. Поэтому решил установить ещё раз ElementaryOS в VirtualBox. Зачем?

Я подумал, что было бы неплохо написать несколько простых уроков для начинающих, используя возможности Linux. Начнём?

Урок первый. Базовые представления об электрических цепях

У меня не слишком хорошее воображение. Я, например, не могу представить четырехмерные геометрические фигуры. По этой причине приходится доверять математике, запоминая формулы и их назначение. Там же, где можно представить какую-то модель происходящего, мне удобно эту модель использовать. Что необязательно для вас.

Начнем с того, что любая электрическая цепь – это соединение радиоэлементов. Если они соединены правильно, то, скорее всего, вы получите полезное устройство. Как правило, электрическая цепь дает возможность проходить по ней постоянному току или переменному току. Начнём с постоянного тока.

Электрический ток – это перемещение электрических зарядов под воздействием источника питания. Пока источник не подключен заряды перемещаются хаотически, а при подключении перемещаются от одного полюса к другому. В технике принято считать, что постоянный ток – это перемещение зарядов от плюса к минусу. Под зарядами мы будем понимать чаще всего электроны, но это могут быть, например, ионы.

Физически это можно представлять себе так: в проводниках есть электроны, слабо связанные с ядром атома, которые легко перемещаются от атома к атому, образуя некое электронное облако. Под действием силы (от источника тока), называемой электродвижущей силой, они движутся упорядоченно. И чем больше электронов проходит в единицу времени через сечение проводника, тем больше сила тока.

На силу тока при заданном значении ЭДС (чаще будем говорить о напряжении) влияет сопротивление, которое проводник оказывает упорядоченному движению электронов. Фактически сопротивление проводников определяется несколькими их параметрами, как то: удельное сопротивление, длина проводника, поперечное сечение проводника. На практике чаще всего мы

имеем дело с готовым элементом, на котором нанесено значение сопротивления, допуск и, далеко не всегда, допустимая мощность рассеяния.

Протекая по проводнику, имеющему сопротивление, ток создаёт на нём падение напряжения. То есть, электрическая цепь из двух последовательно соединённых сопротивлений будет иметь на них падения напряжения, которые в сумме составят значение напряжения источника.

Элементы электрической цепи могут включаться параллельно. Тогда ток в какой-то точке будет разветвляться. Те же два сопротивления, включённые параллельно будут делить общий ток на две ветки. По каждой протекает своя часть общего тока. Перед «возвращением домой» эти две части вновь соединяются. Общий ток будет равен сумме токов в ветвях.

Проведём эти опыты в программе SimulIDE.

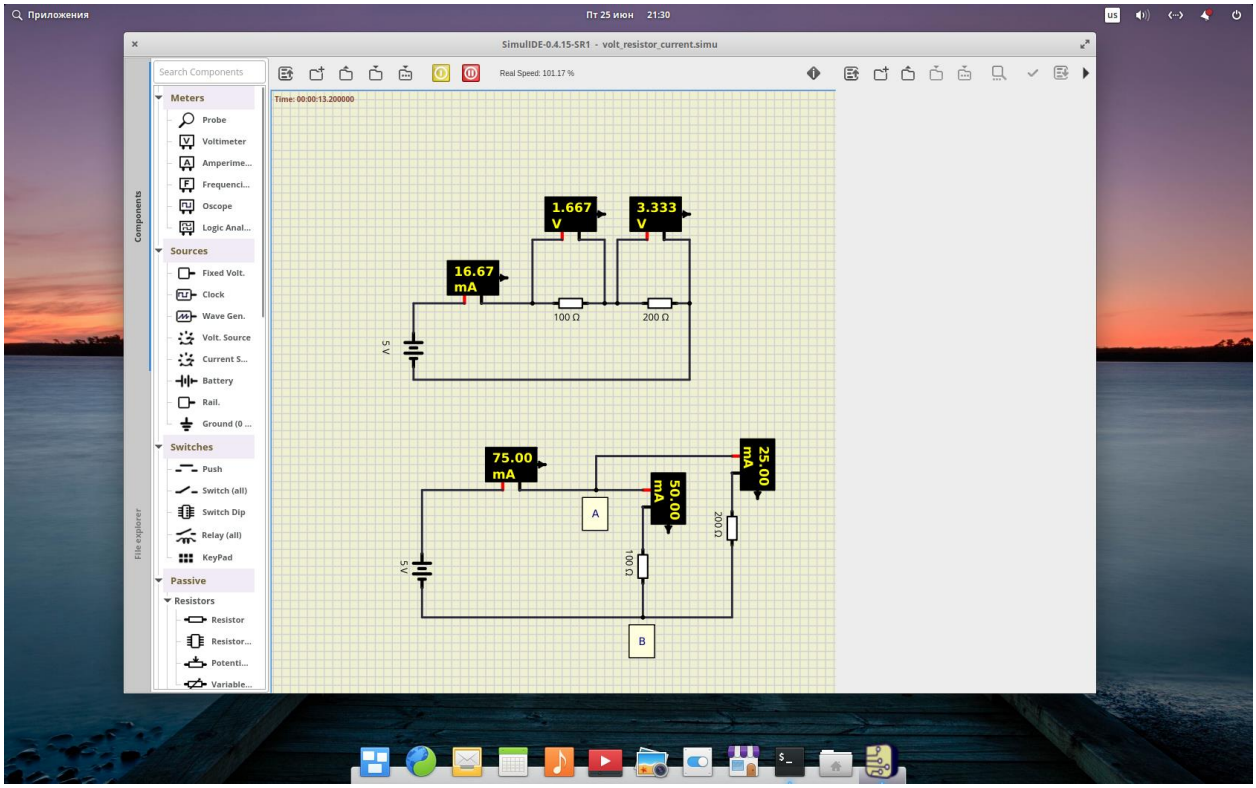


Рис.17.1. Опыты с двумя резисторами на постоянном токе

В первом опыте мы соединили два резистора с разным сопротивлением последовательно. Что можно сказать по результату эксперимента?

1. На первом резисторе, сопротивление которого меньше, меньше падение напряжения. О чём было сказано выше: чем меньше сопротивление, тем меньше падение напряжения.
2. Если сложить оба падения напряжения: 1,667+3,333=5 (В). То есть, ЭДС распределяется в виде падений напряжений на двух резисторах (или большем их количестве).

Между сопротивлением, падением напряжения на сопротивлении и током через него есть простое соотношение, которому мы обязаны учёному по фамилии Ом: U=R*I. Используя основные единицы, то есть: вольт, ом и ампер, - мы можем определить падение напряжения на первом сопротивлении. Амперметр показывает 0,01667А, сопротивление 100 Ом. Результат умножения первого значения на второе будет 1,667 В, что и показывает вольтметр.

Во втором опыте эти же резисторы соединены параллельно. Что можно сказать?

1. Ток от источника ветвится в точке А (напомню, техническое направление тока от плюса к минусу).
2. Через резистор с меньшим сопротивление протекает больший ток, чем через резистор с большим сопротивлением.
3. Сумма токов в ветвях равна току, втекающему в узел А: 50+25=75 (мА).
4. Если включить амперметр между точкой В и минусом источника тока, его величина будет равна 75 мА («ничто в природе не исчезает бесследно, кроме зайцев»).

Из этих опытов можно сделать ещё и такой вывод: при последовательном соединении сопротивлений результирующее сопротивление будет равно сумме значений сопротивлений. Для первой цепи: Rобщ. = 100+200 = 300 (Ом). При этом ток в цепи: 5 (В)/300 (Ом) = 0,01667 (А).

При параллельном соединении: 5 (В)/0,075 (А) = 66,667 (Ом).

Как, зная значения параллельно включённых сопротивлений, определить результирующее сопротивление? При параллельном соединении можно складывать значение проводимостей – величины обратной сопротивлению. В данном случае: 1/Rобщ. = 1/100 + 1/200 = 0,015 (сименс). А сопротивление: Rобщ. = 1/0,015 = 66,667 (Ом). При этом ток, потребляемый от источника тока: 5/66,667 = 0,075 (А).

Каким образом эти два опыта с двумя резисторами могут помочь нам в понимании электрических схем? В первую очередь они помогают нам запомнить закон Ома для участка цепи:

R = U/I (сопротивление участка цепи равно частному от деления падения напряжения на ток)

Или

U = R*I (падение напряжения на участке цепи равно произведению сопротивления на ток)

Или

I = U/R (ток через участок цепи равен частному от деления напряжения на сопротивление)

И запомнить два правила Кирхгофа:

Сумма падений напряжений в цепи равна ЭДС источника и сумма токов в ветвях цепи равна втекающему (или вытекающему) в узел току.

Посмотрим, как эти знания помогут нам продолжить знакомство с электрическими схемами.

Урок второй. Транзистор в электрической цепи

Транзистор, пришедший на смену электронным лампам, по сей день остаётся важнейшим элементом любого электронного устройства. Когда-то транзистор существовал только сам по себе, имея «индивидуальную квартиру», корпус, в котором он выпускался и с которым продолжал жизнь. Сегодня он чаще всего живёт «в общежитии», в корпусе микросхемы. И жить там могут сотни, тысячи и миллионы транзисторов.

Тем не менее, каждый из транзисторов остаётся сам собой. Биполярный транзистор – это токовый элемент, у которого ток коллектора зависит от тока базы, что определяется простым соотношением: Iк = Bст*Iб (ток коллектора в «В» раз больше тока базы, где «В» просто число).

История появления полупроводниковых приборов сама по себе очень поучительна. Когда-то, исследуя свойства разных материалов (с точки зрения электричества), остановились на двух: проводники и изоляторы. Первые хорошо проводили ток, вторые почти не проводили ток. Те материалы, которые проводили ток, но плохо, отбраковали (полупроводники).

Но последующие исследования показали, что есть два типа полупроводниковых материалов – их назвали полупроводники n-типа и p-типа. Когда соединили две пластины из полупроводников разного типа проводимости, обнаружили, что полученная конструкция вела себя подобно двухэлектродной электронной лампе, диоду, при одном приложении напряжения ток пропускала, при другом нет. Так появился полупроводниковый диод. За счёт чего появился этот эффект?

При соединении полупроводников на границе раздела произошло следующее: электроны из полупроводника n-типа перешли границу раздела и осели на атомах полупроводника p-типа. Образовалась заряженная пограничная область. Она-то и придала новой конструкции свойства диода.

Комбинация из трёх полупроводников разного типа проводимости дала два типа транзисторов: биполярные транзисторы n-p-n типа и p-n-p типа. Конструкция напоминает сэндвич: между двумя полупроводниковыми областями одного типа расположен полупроводник другого типа.

Проведём такой опыт:

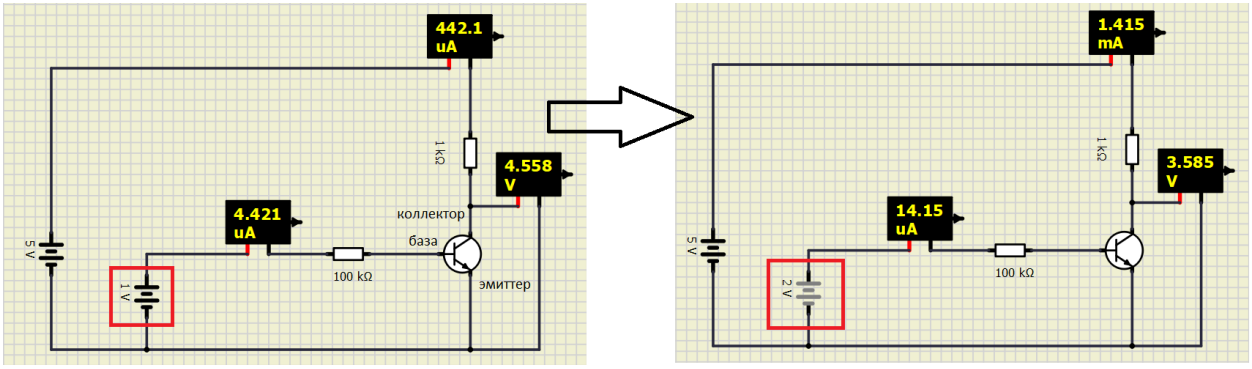


Рис. 17.2. Опыт с биполярным транзистором n-p-n типа

Схемы этого эксперимента идентичны, исключая напряжение батареек в цепи базы транзистора.

Кстати, транзистор – это трёх выводное устройство. У биполярного транзистора n-p-n различают эмиттер (здесь он соединён с минусом батарейки), базу (она соединена с полупроводником p-типа) и коллектор (он соединён с резистором нагрузки). Для транзистора p-n-p типа те же электроды, но база соединена с полупроводником n-типа.

Напряжение батарейки в базовой цепи определяет ток базы. В первом случае ток базы 4,421 микроампер, во втором – 14,15 микроампер. Разделив ток коллектора на ток базы, мы в обоих случаях получим 100. Это тот коэффициент B, который определяет соотношение между током базы и током коллектора.

Любой датчик, способный генерировать напряжение, подключённый к базовой цепи, задаёт ток базы. Когда меняется что-то, на что реагирует датчик, меняется его напряжение, меняется ток базы, но ток коллектора меняется в 100 раз больше. То есть, транзистор усиливает ток, что от него и требуется в качестве активного элемента – усилителя тока. Но часто требуется усилить напряжение. Как это получается?

Если мы включим вольтметр между базой и эмиттером транзистора, то напряжение в первом случае будет 558 мВ, во втором 584 мВ. То есть, изменение напряжения 26 мВ. При этом напряжение на коллекторе транзистора изменилось на 973 мВ. Это отношение напряжений и покажет усиление напряжения. Проверьте, так ли это?

Биполярный транзистор p-n-p типа даст такие же результаты эксперимента, но полярность батареек нужно будет изменить.

В этом эксперименте мы использовали две батарейки, но это слишком «расточительно» для практического использования. Если у вас с десяток транзисторов, то всё устройство будет сплошным «общежитием» батареек.

На практике поступают иначе. Ток базы задают, включая резистор между базой транзистора и, например, полюсом источника. Вот так:

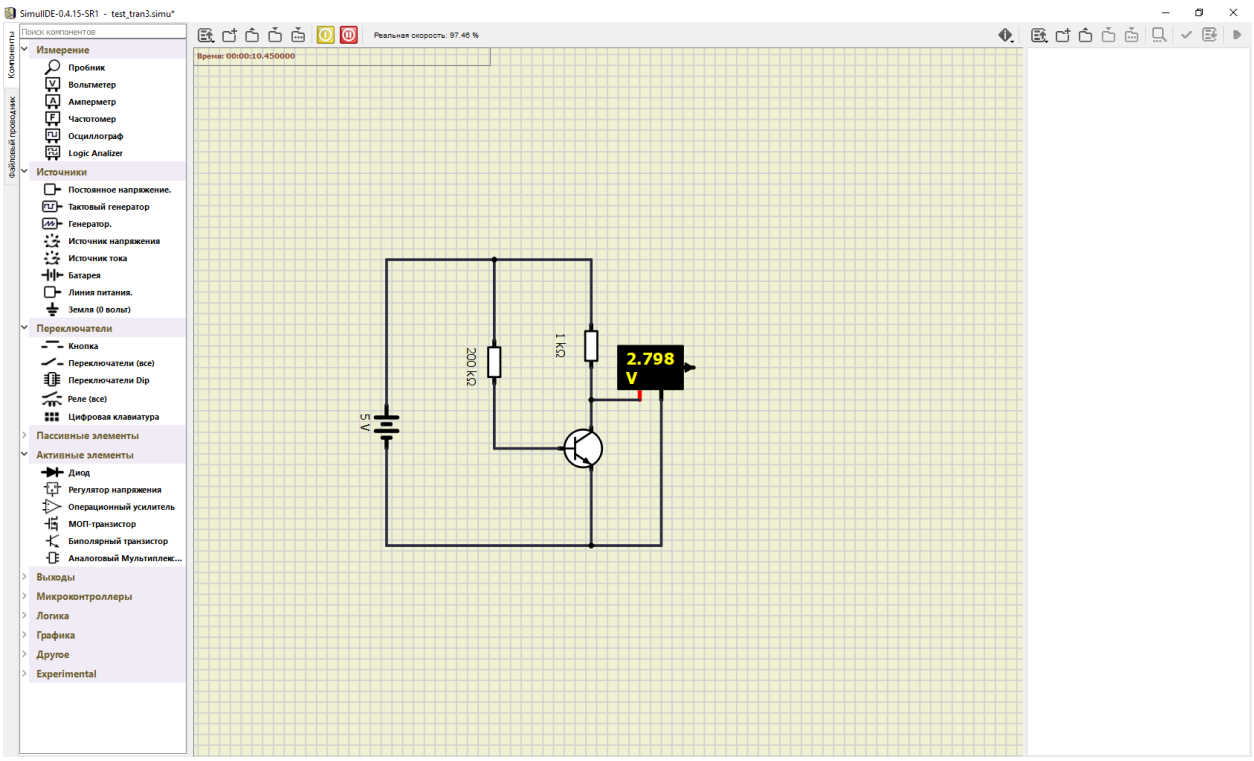


Рис. 17.3. Включение сопротивления для задания тока базы

В зависимости от характера напряжения, которое требуется усиливать, выбирают такую величину резистора в цепи базы, которая обеспечивает нужное напряжение на коллекторе транзистора (падение напряжения эмиттер-коллектор в этой ветви схемы). Для симметричной формы входного напряжения желательно иметь напряжение на коллекторе транзистора равное половине напряжения питания.

Всё вышеизложенное относилось к биполярным транзисторам. Но есть ещё один тип транзисторов, которые называют полевыми или канальным транзисторами. В чём различие?

Различие в конструкции. У полевого транзистора основа – это полупроводник одного типа, имеющий два вывода. Один вывод называют истоком, другой называют стоком. «По бокам» этого полупроводника расположены области полупроводника другого типа, выводы которого объединены в вывод, который называют затвором. Что происходит в этом случае?

Как и у биполярного транзистора на границах соединения материалов появляются заряженные зоны, они мешают протеканию тока от истока к стоку. Если к затвору приложить напряжение, то эти зоны могут расширятся, уменьшая ток через оставшийся канал. То есть, ток от истока к стоку зависит от напряжения на затворе канального (или полевого, из-за образованных приграничных полей) транзистора.

Полевой транзистор больше похож на электронную лампу (триод), у которой редко используется режим с сеточным током. У полевого транзистора ток через затвор не протекает. Управляется полевой транзистор напряжением между истоком и затвором. Проведём такой эксперимент.

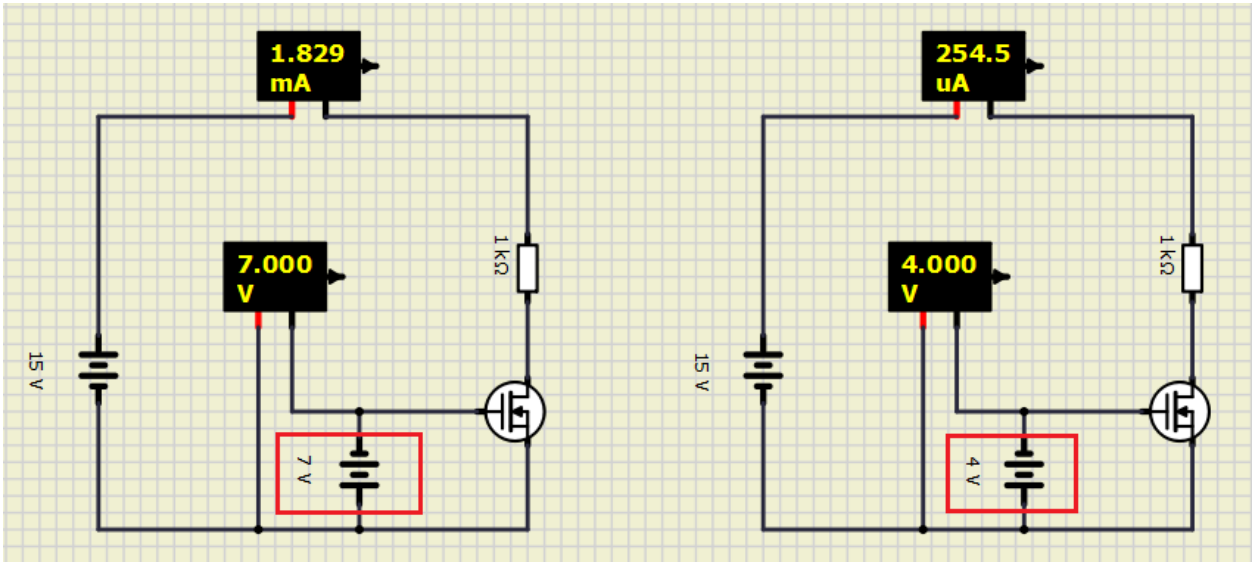


Рис. 17.4. Эксперимент с полевым (канальным) транзистором

Изменение напряжения на затворе (7 вольт в первом случае) меняет ток между истоком и стоком (1,829 мА в первом случае и 254,5 мкА во втором). Как и в случае задания начальных условий работы транзистора для биполярной модели, так и для полевого транзистора применяют резистор, определяющий заданный ток стока. Падение напряжения на резисторе в цепи истока приложено к затвору через резистор с большим сопротивлением. Полярность этого напряжения оказывается такой, что транзистор открывается.

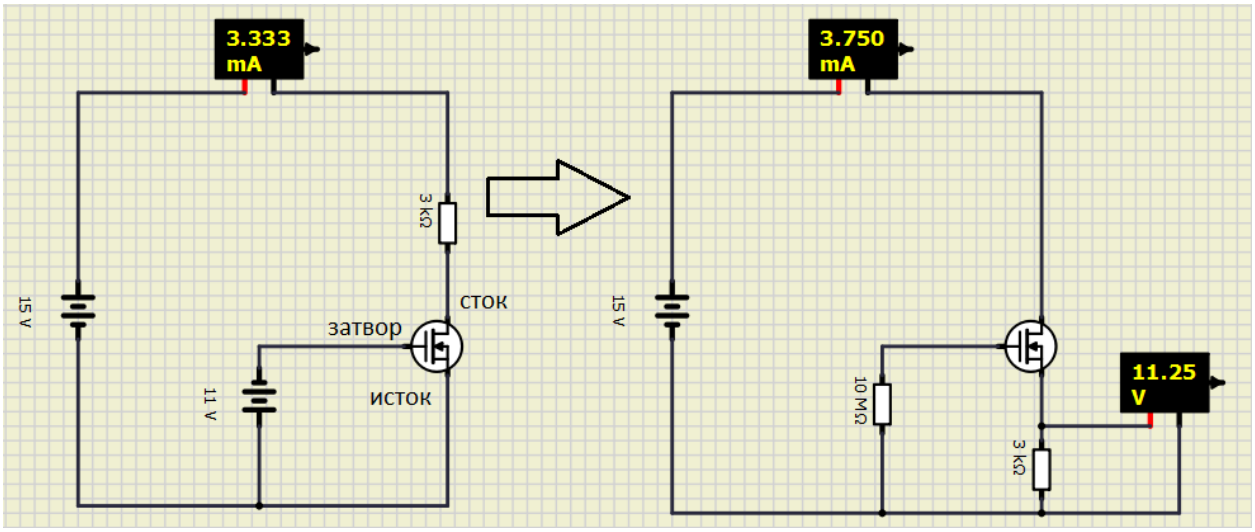


Рис. 17.5. Начальные условия работы полевого транзистора

Мы познакомились с транзистором. Как мы использовали (или не использовали?) ранее полученное представление о напряжении, токе и сопротивлении? Начнём с последнего рисунка.

На схеме слева мы получили ток стока 3,333 мА при напряжении на затворе -11 вольт. Зададимся этими напряжением и током для получения значения сопротивления в цепи истока: 11/0.003333 =

3300 (Ом). Добавив в цепь истока сопротивление 3 кОм, мы получили ток стока 3,75 мА, что очень близко к требуемой величине.

Обратимся к рис. 17.3, чтобы понять, как мы выбирали значение резистора в цепи базы. Мы намеревались получить напряжение на коллекторе транзистора близким к 2,5 В (половина питающего напряжения). Но это означает, что на резисторе в цепи коллектора тоже падает напряжение 2,5 В. Разделим это напряжение на сопротивление нагрузки (сопротивление в цепи коллектора), чтобы найти ток коллектора: 2,5/1000 = 0,0025 (А). Ранее мы определили значение Вст, которое составило 100. То есть, ток базы в 100 раз меньше полученного нами значения тока коллектора. Но напряжение эмиттер-база порядка 0,5-0,7 В (переход база-эмиттер, в сущности, имеет свойства диода, на котором при прямом включении падает не больше). Из этого следует, что на сопротивлении в цепи базы будет падать напряжение порядка 4,5 В. Мы знаем напряжение, мы знаем ток (напомню, 0,000025 А). Что мешает нам определить сопротивление?

4,5В/0,000025А = 180 000 Ом

Мы выбрали 200 кОм, получив напряжение на коллекторе 2,798 В. Полученные нами представления о напряжении, токе и сопротивлении помогли нам рассчитать каскад усиления на транзисторе.

Урок третий. Переменный электрический ток

Когда мы говорили о постоянном электрическом токе, то представляли его как упорядоченное движение электрических зарядов. Но мы не уточнили, что упорядочение может иметь разный характер. Для постоянного тока характерно, что он не меняется ни по величине, ни по направлению. Иначе будет с переменным электрическим током.

С переменным током, который изменяется по разным законам, приходится работать в электронных устройствах. При изучении работы на переменном токе удобно пользоваться синусоидальным электрическим переменным током. То есть, током, величина которого изменяется во времени по закону синуса. Используем этот ток для получения представления об усилении транзистора по напряжению в схеме усилителя, где транзистор включён с общим эмиттером… Стоп, стоп, стоп!

Начнём мы не с этого. Начнём с повторения опыта на рис. 17.1, где заменим батарейку источником переменного напряжения. Начнём:

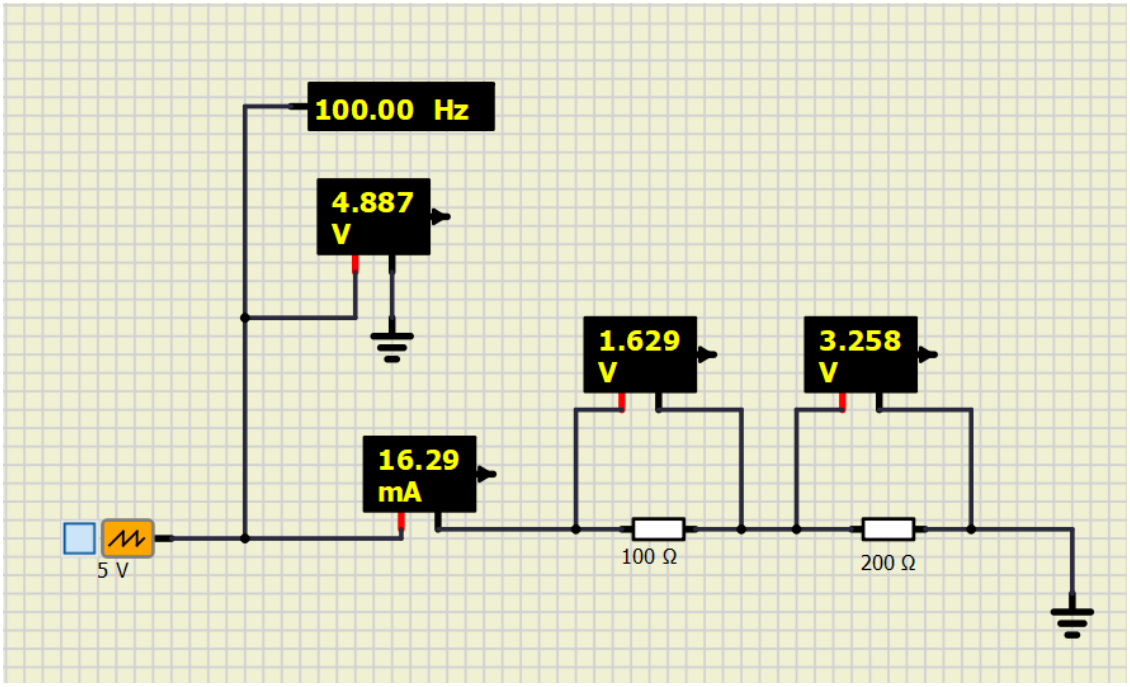


Рис. 17.6. Опыт с двумя резисторами и источником переменного напряжения

Если бы мы не знали, какой источник напряжения подключён к цепи из двух резисторов, мы не могли бы сказать, с постоянным или с переменным током мы имеем дело. Показания приборов очень похожи. Почему это происходит?

Для переменного синусоидального тока имеет значение амплитуда (как функции синуса), но на практике говорят о действующем значении переменного тока (напряжения), которое меньше амплитудного значения в 1,41 раз. При таком значении переменного тока он оказывает такое же действие на активное сопротивление, как и постоянный ток. Лампочка будет одинаково светить и при постоянном, и при переменном токе того же действующего значения.

Закон Ома для участка цепи при переменном токе будет справедлив в том же виде, что и для постоянного тока.

Рассматривая постоянный электрический ток, мы не упоминали два элемента, которые (можно это утверждать) присутствуют во всех современных электронных устройствах. Даже если вы не увидите их на принципиальной схеме, уверяю вас – они будут присутствовать. Это конденсатор и индуктивность. Конденсатор можно представить в виде двух металлических пластин, между которыми есть изолятор. Постоянный ток такая конструкция не пропускает.

Индуктивность представляем себе в виде катушки, на которую наматывается провод. Если провод толстый, то сопротивление постоянному току очень маленькое (в идеале равное нулю).

Для переменного тока дело обстоит иначе. У переменного тока кроме амплитуды есть ещё один важный параметр – частота. К чему это я?

К тому, что на переменном токе и индуктивность, и конденсатор имеют сопротивление, которое зависит от частоты переменного тока. Изначально я хотел показать простые опыты в SimulIDE для этих элементов, но у симулятора происходит какой-то раскардаш в электрических цепях, которые я хотел показать. Пришлось изменить подход. Вот как всё выглядит для конденсатора.

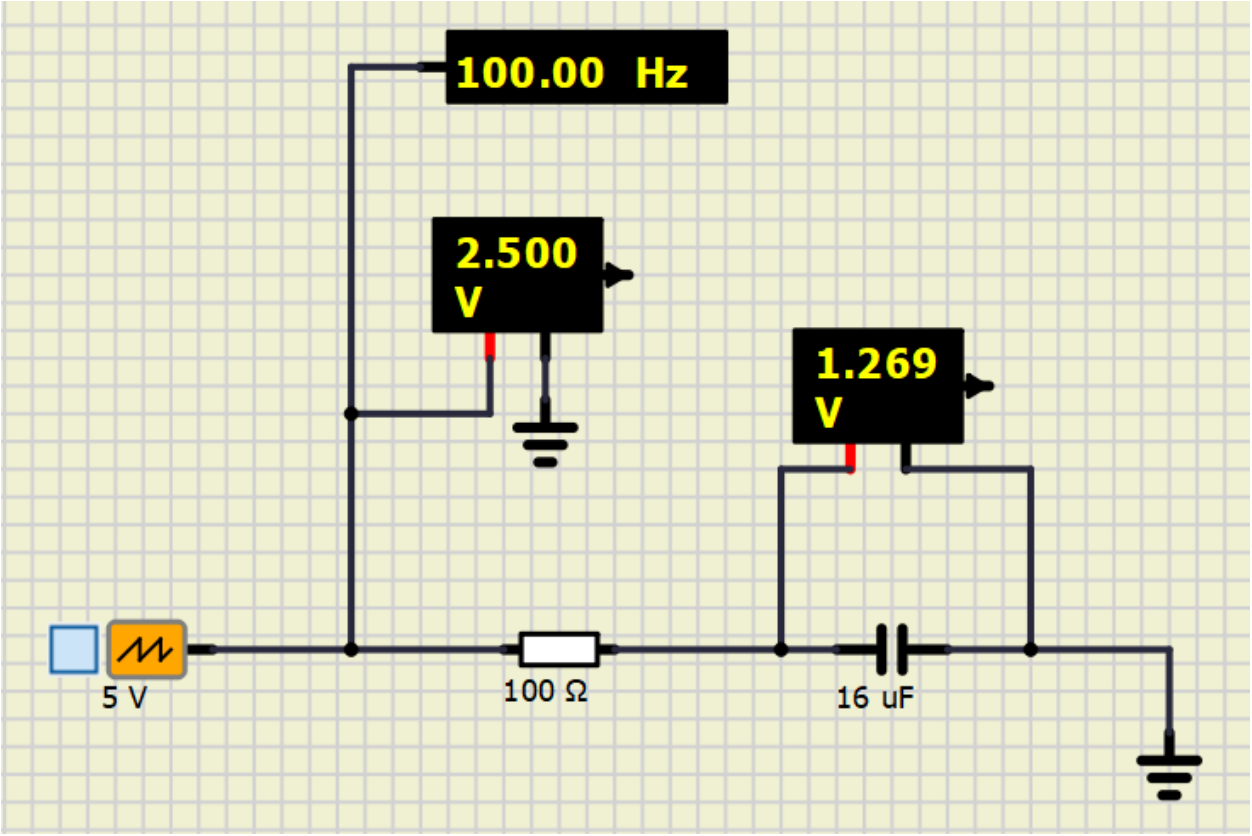


Рис. 17.7. Схема для определения сопротивления конденсатора

Напряжение в 2,5 В от генератора переменного напряжения с частотой 100 Гц делится на два между резистором и конденсатором, что предполагает равенство их сопротивлений. То есть, конденсатор с ёмкостью 16 мкФ на частоте 100 Гц синусоидального напряжения имеет сопротивление 100 Ом.

Я не знаю, какое напряжение генератора задаётся, амплитудное или действующее. Я не знаю, как измеряется это напряжение вольтметром – это вольтметр постоянного тока или переменного? Тем не менее, повторим эту схему для индуктивности.

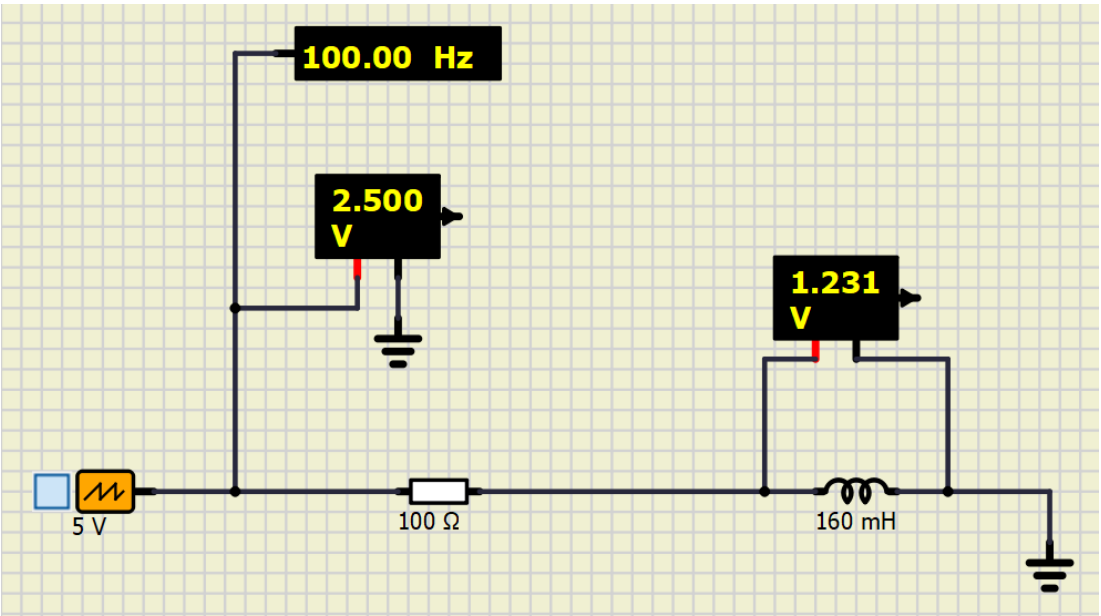


Рис. 17.8. Схема для определения сопротивления катушки индуктивности

Отчего-то показания могут измениться, когда делаешь снимок экрана. Впрочем, эту часть рассказа я делаю полностью в Windows, что получится в Linux нужно проверить (любопытства ради).

Сопротивления, которые мне хотелось показать, называются реактивными емкостным и индуктивным сопротивлением. Определяются они формулами:

Xc = 1/2πfC, где частота f герцах, а ёмкость C в фарадах

X_L = 2πfL, где частота в герцах, а индуктивность генри

И хотелось бы посмотреть, что же с генератором и вольтметрами?

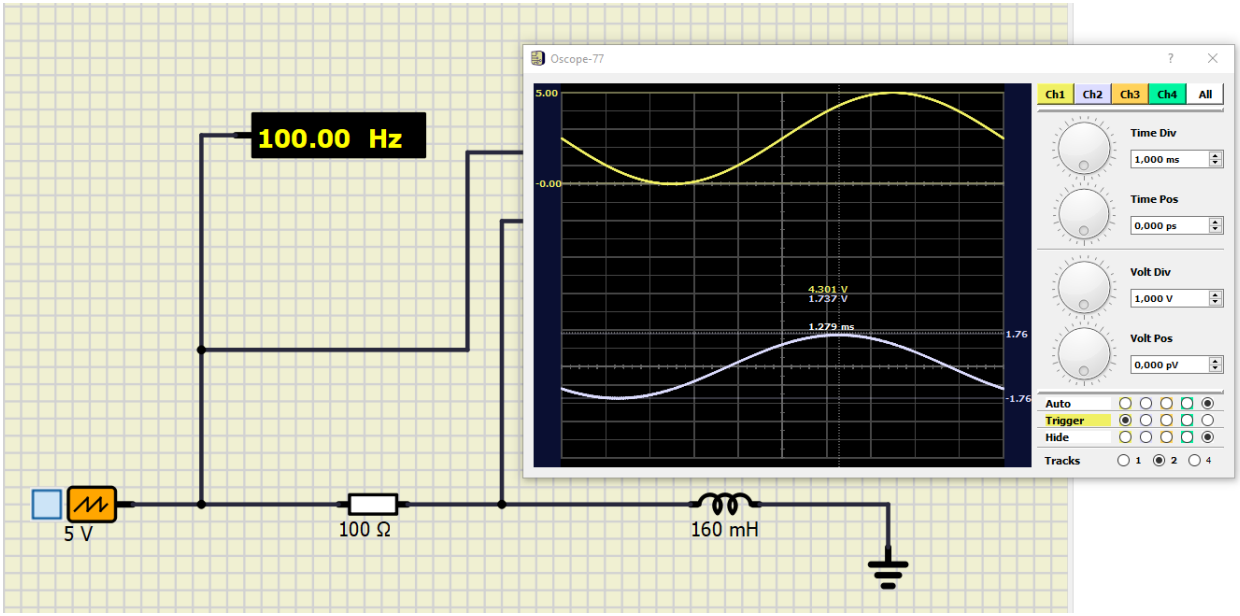


Рис. 17.9. Подключение осциллографа к электрической цепи рис. 17.8

Получается, что 5 В, задаваемые для генератора – это двойная амплитуда напряжения. В этом случае вольтметр показывает просто амплитуду.

И наконец, а что нам скажет ElementaryOS?...

И скажет эта ОС то же, что и Windows. Но, если сместить напряжение генератора так, чтобы ток менял не только величину, но и направление, становится понятно, что амперметр и вольтметр предназначены для измерения постоянного напряжения. Вот вам и раскардаш!

Урок четвёртый. Однокаскадный усилитель на транзисторе

Любой усилитель имеет вход и выход. Как правило, не исключаю, что есть исключения, к входу подключают датчик. Датчиком может служить, например, микрофон или головка звукоснимателя (магнитофона), датчиком может служить фотоэлемент или термоэлемент. Словом, на вход подают входной сигнал, который предстоит усилить, чтобы на выходе получить более мощный сигнал. Мы пока не определили, что называть сигналом, но достаточно пока представлять его в виде некоторого напряжения, которое меняется, чтобы передать нам что-то, какую-то информацию.

Мы будем рассматривать входной сигнал в виде небольшого переменного напряжения синусоидальной формы. Синусоидальное напряжение в качестве испытательного сигнала используют очень часто.

Предполагая усилить сигнал с помощью транзистора, мы можем его включить с общей базой, с общим коллектором и общим эмиттером. «Общий» означает, что это общая точка подключения

входного и выходного напряжения. Часто «общий провод» подключают к «земле». То есть, на схеме графическим элементом «заземление» обозначают общий провод, относительно которого проводят измерения.

Мы рассмотрим схему с общим эмиттером, как наиболее часто используемую. Причина в том, что включение транзистора с общим коллектором имеет усиление по напряжению не более единицы, а включение с общей базой имеет меньше единицы усиление по току. Только включение с общим эмиттером имеет усиление больше единицы и для напряжения, и для тока.

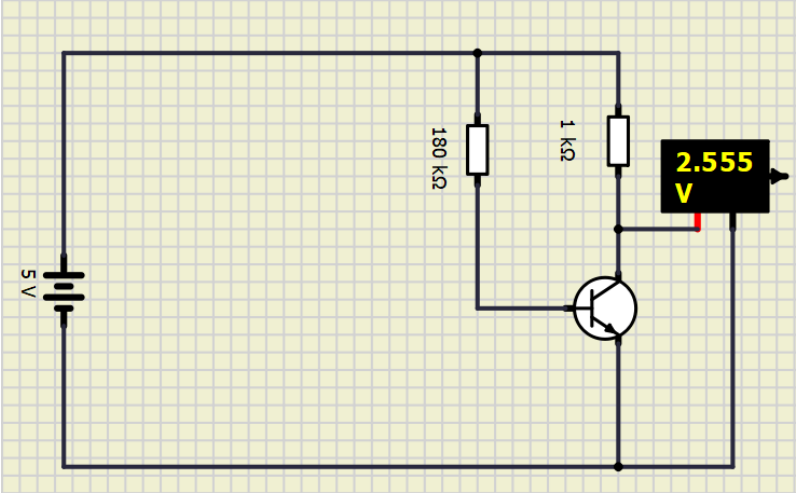


Рис. 17.10. Однокаскадный усилитель на транзисторе, выбор рабочей точки

Выше было сказано, что для симметричных сигналов (входных напряжений), а синусоидальный сигнал симметричен, желательно выбрать такой режим, при котором напряжение на коллекторе равно половине напряжения питания.

Теперь подключим к входу генератор, а к выходу осциллограф.

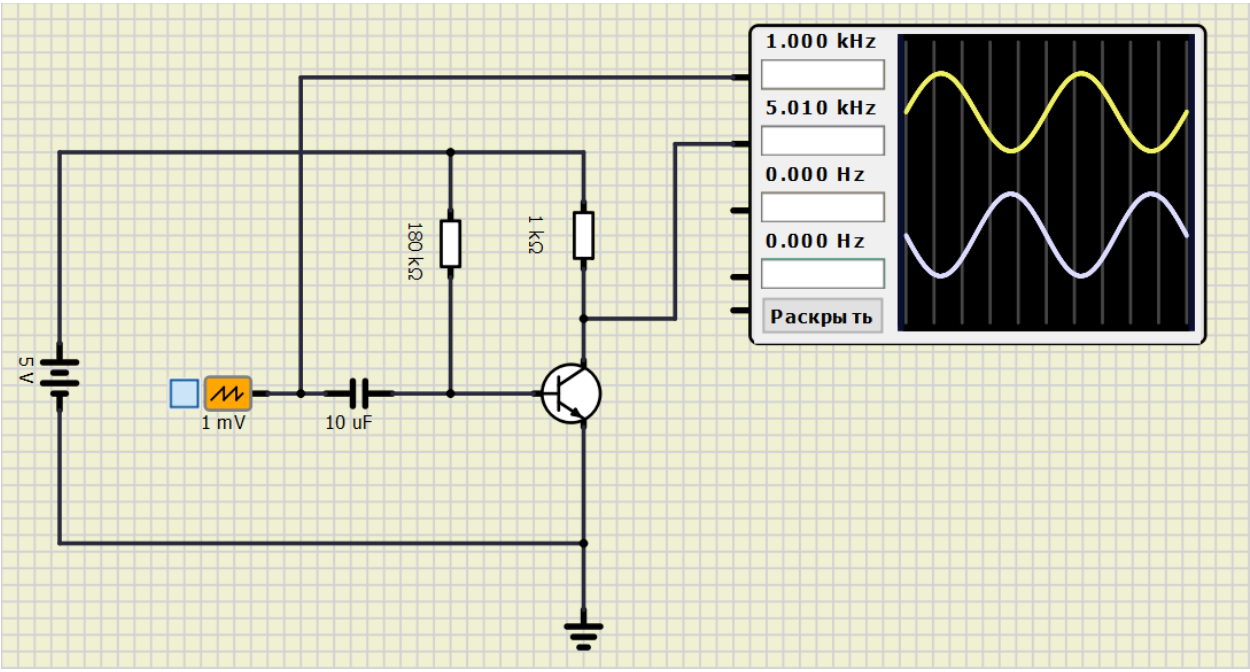


Рис. 17.11. Подключение генератора и осциллографа для проведения измерений

Входной сигнал мы получаем от генератора с заданным напряжением порядка 1 мВ. На выходе напряжение фиксируется осциллографом (как и на входе, впрочем). Из рисунка трудно понять, что второй провод генератора и второй провод осциллографа подключены к «земле» (к общему проводу схемы). То есть, эмиттер является общим и для входного, и для выходного сигнала.

Что про усиление напряжения? Хотя осциллограф в SimulIDE работает с маркерами, с ними не всё понятно. Поэтому можно поступить иначе:

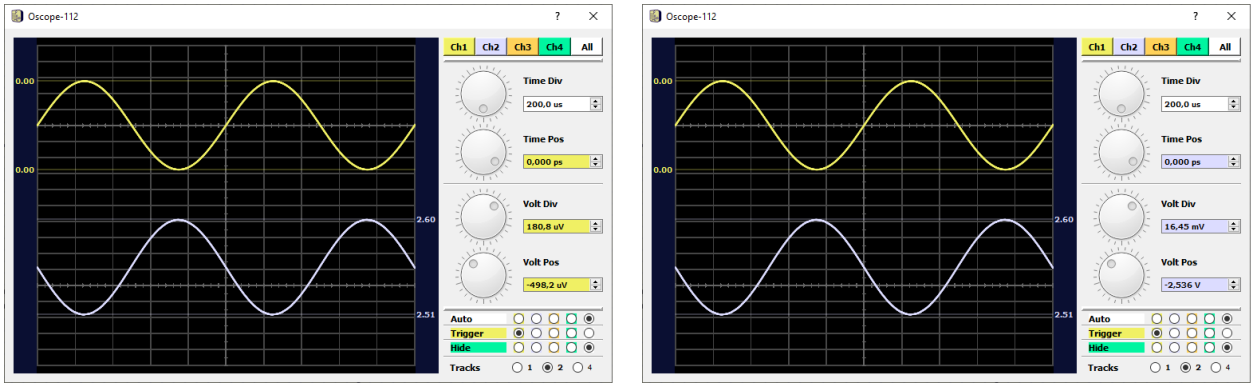


Рис. 17.12. Определение величины сигнала на входе и на выходе

На экране оба сигнала имеют равную амплитуду. Переключателем в верхней части можно выбрать канал. На первом снимке экрана выбран первый канал с входным сигналом. Чувствительность канала порядка 180 мкВ на деление. Посчитав деления от нижней части синусоиды до верхней, можно сказать, что двойная амплитуда сигнала порядка 1,08 мВ (180 мкВ*6). Переключив теперь настройки на второй канал (выходное напряжение), можно определить по чувствительности 16 мВ, что двойная амплитуда порядка 98 мВ. Отношение выходного напряжения к входному: 98/1 = 98, - даёт усиление по напряжению. При этом заметных искажений выходного сигнала не видно.

Кроме усиления по напряжению нас может интересовать ряд других параметров. В первую очередь хотелось бы получить АЧХ (амплитудно-частотную характеристику). Во многих случаях мы должны быть уверены, что усилитель усиливает определённый спектр частот. Простейший случай – это усилитель звука. Если мы хотим усилить речь, то достаточно частотного диапазона порядка 300-3000 Гц. Для усиления музыки желательно иметь диапазон частот 30-15000 Гц. В зависимости от назначения усилителя выбирается транзистор, который обеспечивает этот параметр.

В конкретном случае нижний диапазон частот определяется ёмкостью входного конденсатора и входным сопротивлением каскада. Верхний предел частот зависит во многом от транзистора. Верхняя частота 15 кГц в данном случае достигается при применении любого современного транзистора.

Для получения АЧХ мы используем другую программу, которая тоже работает в среде ElementaryOS, QucsStudio.

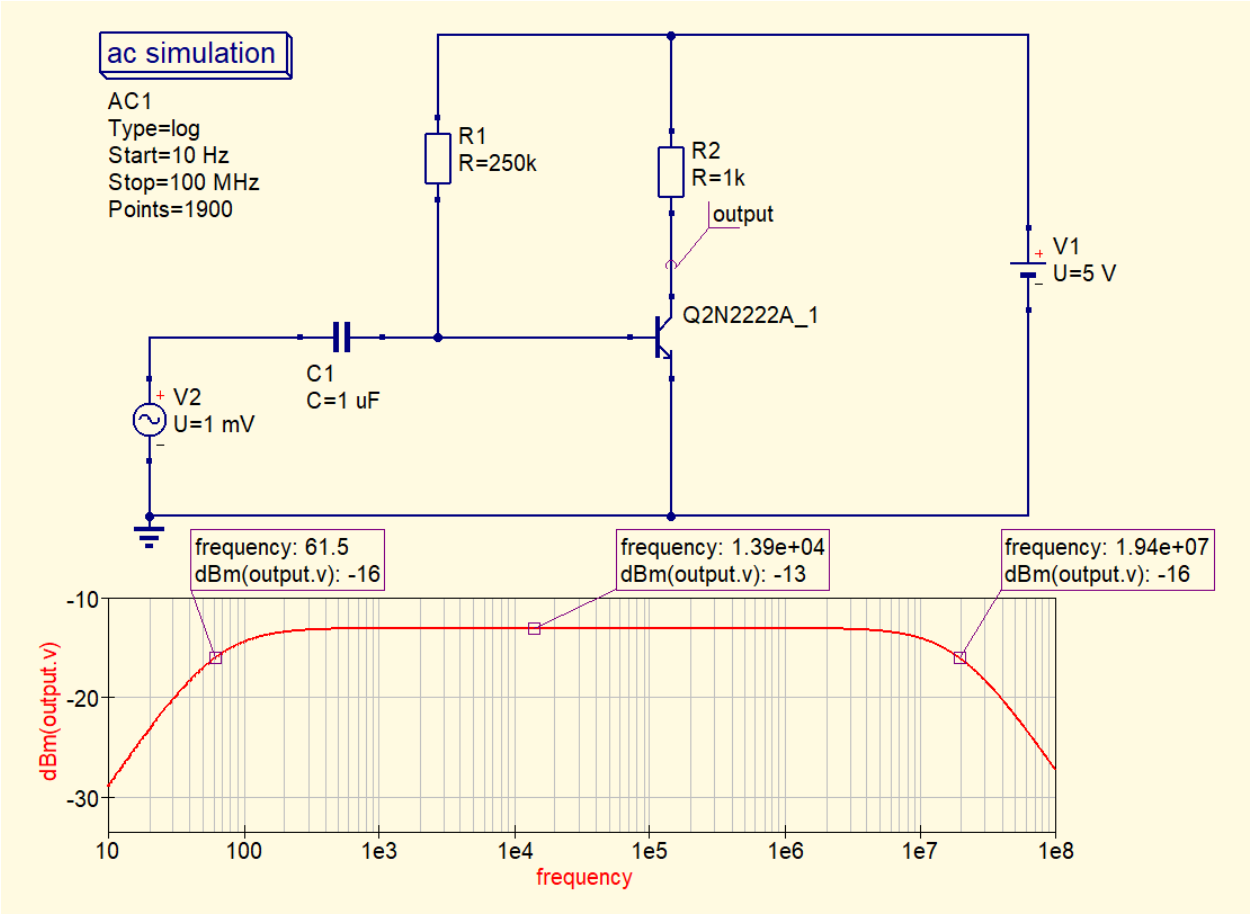


Рис. 17.13. Амплитудно-частотная характеристика усилителя с транзистором 2N2222A

В схеме есть изменения – резистор в цепи базы имеет сопротивление 250 кОм, что связано с выбором рабочей точки для данного транзистора. Входной конденсатор имеет ёмкость 1 мкФ, что связано с желанием получить АЧХ на низких частотах, на которой можно отметить нижнюю рабочую частоту.

Теперь о самой АЧХ. Частотную характеристику удобно строить в децибелах. По соглашению в качестве граничных частот выбираются те, на которых частотная характеристика спадает на 3 дБ. В центральной части выходной сигнал -13 дБ, а по краям диапазона -16 дБ. Таким образом, нижняя рабочая частота порядка 60 Гц, а верхняя порядка 20 МГц.

В программе SimulIDE транзистор, скорее всего идеальный, для которого трудно определить верхнюю граничную частоту. Иначе мы могли бы менять частоту генератора до тех пор, пока сигнал не уменьшится в 1,41 раз. Так можно определять граничные частоты на практике, если есть генератор и вольтметр переменного тока, работающий в широком диапазоне частот.

Децибелы – это относительные единицы, определяемые для усиления по напряжению формулой:

Кдб = 20log(Uвых/Uвх)

Использование децибел удобно тогда, когда, например, у вас два каскада усиления. Общее усиление определится сложением усиления каждого каскада в децибелах.

Ранее мы определили коэффициент усиления каскада по напряжению, вычислив, что он равен 98. В децибелах это будет около 40 дБ (легко вычислить на калькуляторе).

При выборе транзистора для усилителя мы обратили внимание на его частотные свойства, но во многих случаях требуется проверить, не превышает ли ток коллектора, напряжение на коллекторе, мощность, рассеиваемая на коллекторе, предельно допустимых значений. Сделать это удобно, используя программу моделирования. Если даже вы не получите абсолютно точных значений, то полученные данные позволят вам аккуратнее проверить эти параметры на макетной плате. Транзистор можно быстро вывести из строя, если не учитывать все его возможности.

Для усилителей, как предварительных, так и оконечных, важное значение играют вносимые ими искажения. Мы рассчитали рабочую точку транзистора. Мы выбрали её из соображений получения максимального напряжения на выходе усилителя. Но сделали мы это только для одного случая температурного режима. Положим, что усилитель, о котором речь шла выше, будет работать в устройстве, предназначенном для улицы. А на улице температура может меняться от – 20 до + 30 градусов. Между тем полупроводниковые приборы очень чувствительны к температуре. Так как мы можем «защитить» выбранную рабочую точку?

Для стабилизации рабочей точки транзистора применяют отрицательную обратную связь. Сущность обратной связи в том, что к входному сигналу добавляется часть выходного. Если эта добавка вычитается из входного сигнала, говорят об отрицательной обратной связи, иначе обратная связь будет положительной. А положительная обратная связь превратит усилитель в генератор.

Добавим в каскад усиления резистивный делитель, который стабилизировал бы напряжение на базе транзистора. При этом сопротивление нижней части делителя должно быть достаточно небольшим. И добавим сопротивление в цепь эмиттера транзистора. Это сопротивление будет осуществлять отрицательную обратную связь. Как это происходит?

Положим, что из-за нагрева транзистора у него возрастает неуправляемы сквозной ток (есть такой), в этом случае на резисторе в цепи эмиттера возрастает падение напряжения. Но относительно базы это падение напряжения будет вычитаться из напряжения, задаваемого резистивным делителем, то есть, напряжение на базе транзистора (относительно эмиттера) будет уменьшаться, что приведёт к уменьшению тока коллектора (и эмиттера) транзистора, возвращая рабочую точку к первоначальному значению.

Добавление резистора в цепь эмиттера вносит последовательную обратную связь (сигнал обратной связи будет включаться последовательно входному напряжению). Можно использовать и параллельную обратную связь, если подать часть выходного напряжения с помощью резистора, включённого между коллектором и базой транзистора. Вы можете увидеть эти решения на многих схемах электронных устройств. Кроме стабилизации рабочей точки отрицательная обратная связь улучшает ряд других параметров: снижает уровень шумов, уменьшает нелинейные искажения, расширяет рабочую полосу частот усилителя.

Для однокаскадного усилителя отрицательная обратная связь может быть очень глубокой, что не приводит к появлению проблем. Так при включении транзистора с общим коллектором осуществляется стопроцентная отрицательная обратная связь, и какие-либо проблемы с устойчивой работой не появляются. Иначе обстоит дело с многокаскадными усилителями. Для этого случая выбор глубины отрицательной обратной связи представляет отдельную задачу, которую можно и нужно решать.

Причина возникновения проблем с глубокой отрицательной обратной связью в том, что на некоторых частотах в многокаскадном усилителе обратная связь может стать положительной из-за фазовых сдвигов. На рис. 17.12 можно явно увидеть, что выходной сигнал и входной сигнал находятся в противофазе: когда входной сигнал увеличивается, выходной сигнал уменьшается. За

пределами верхней граничной частоты каскада это соотношение нарушается. Но для одного каскада усиления выходной сигнал станет синфазным с входным только при усилении меньше единицы, что не скажешь про многокаскадный усилитель, способный сохранить усиление на этих частотах больше единицы.

Урок пятый. Импульсный режим работы биполярного транзистора

Довольно часто транзистор используют для усиления сигнала, передающего два состояния: высокое значение напряжения на выходе и низкое. Вот, как это выглядит на экране осциллографа:

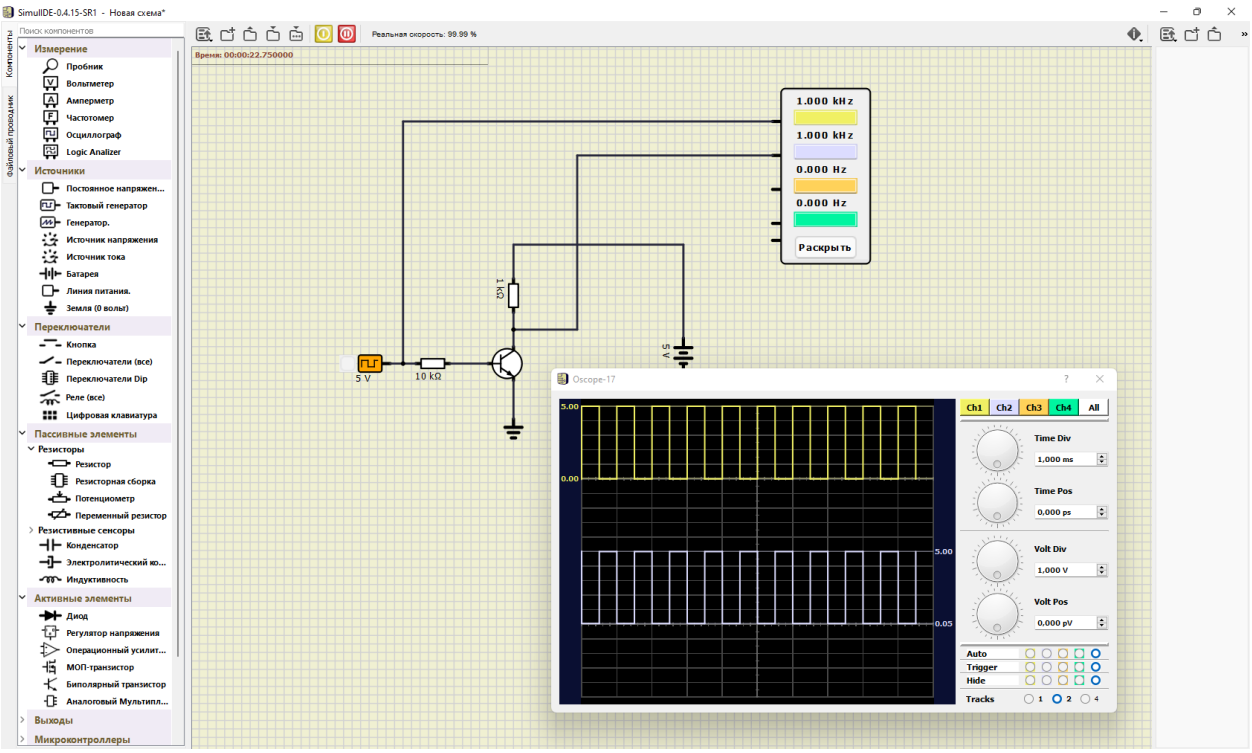


Рис. 17.14. Ключевой режим работы транзистора

И входной сигнал меняется от 0 до +5 В, и выходной сигнал меняется в том же диапазоне. Такая работа транзисторов очень характерна для цифровых устройств. Высокое значение напряжения – это единица, а низкое – ноль.

Для усилительного каскада ранее мы начали рассмотрение с частотных свойств транзистора. Какую роль частотные свойства транзистора играют в ключевом режиме работы транзистора?

Выполним моделирование в программе QucsStudio, где проще получить нужные осциллограммы. Вот выходной сигнал для транзистора в ключевом режиме работы. Генератор импульсов V2 работает на частоте 1 кГц. Осциллограмма представляет один импульс.

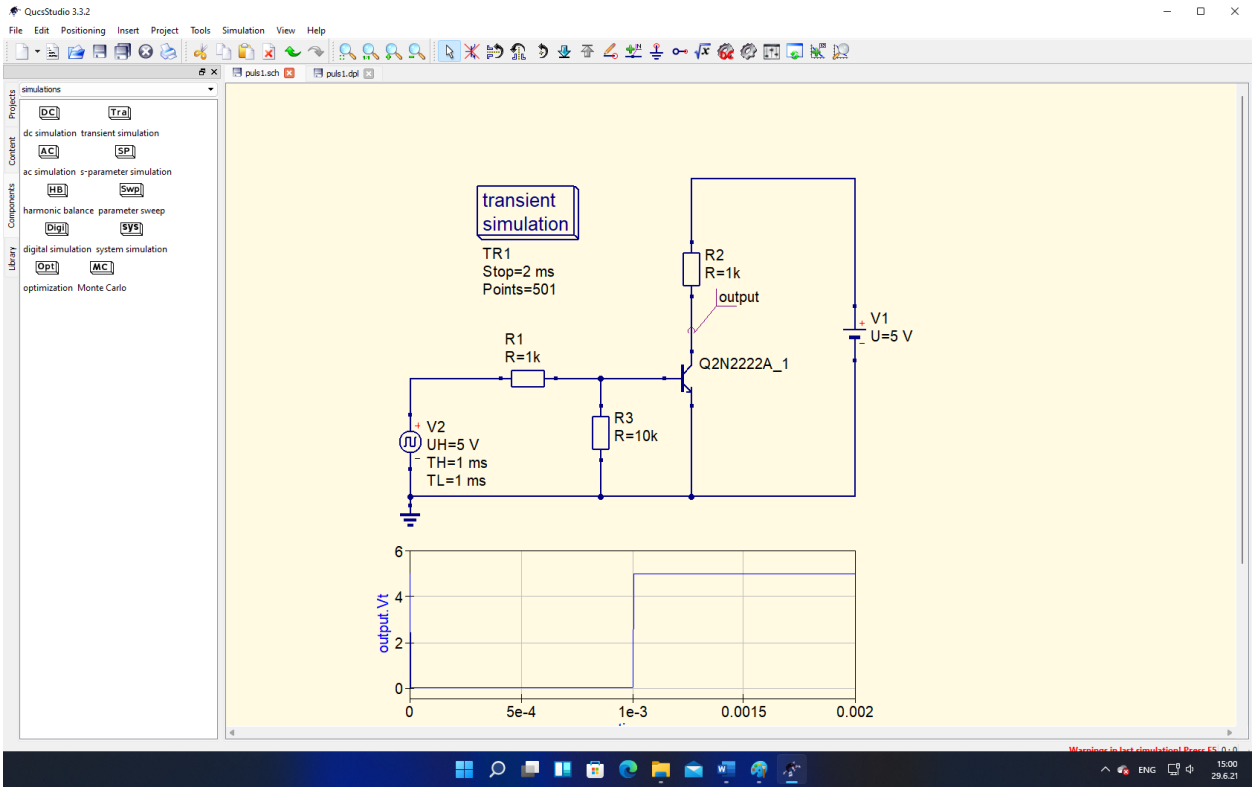


Рис. 17.15. Ключевой режим работы транзистора, моделирование в QucsStudio

В диапазоне отображения сигнала импульс выглядит так, как о нём говорилось раньше – переходит от нулевого напряжения к напряжению 5 вольт. Но, если мы посмотрим не него более пристально…

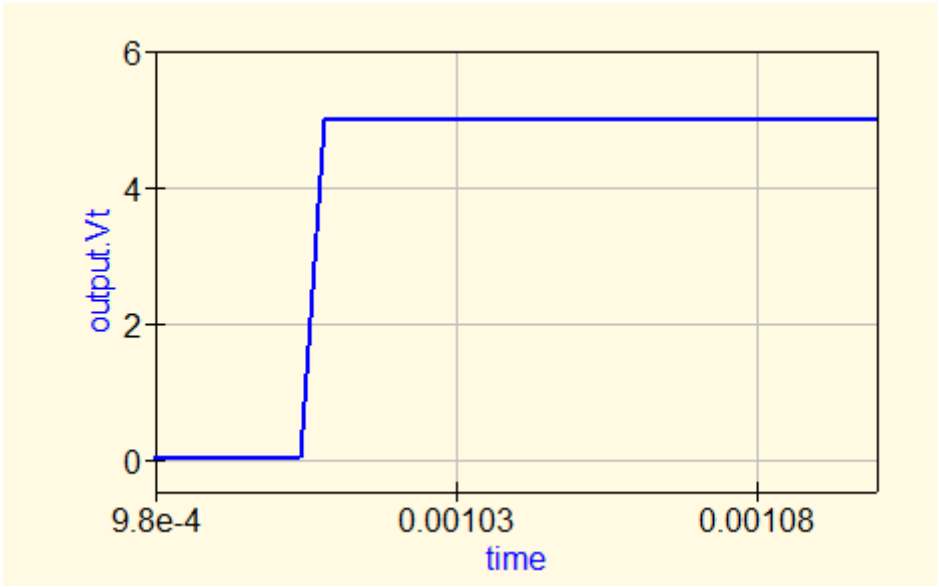


Рис. 17.16. Передний фронт импульса сигнала на выходе

Рассматривая схему, мы предполагаем, что входной сигнал сразу передаётся на выход каскада, но его небольшая задержка может привести к ошибке в работе устройства. Кроме того, если мы рассматриваем мощность рассеивания на коллекторе транзистора без учёта частотных свойств, то рассуждаем так: когда на выходе низкий уровень напряжения, то мощность определяется током коллектора и падением напряжения на полностью открытом транзисторе (напряжением

насыщения), которые малы, и мощность рассеяния невелика. Напряжение насыщения, к слову, это такое напряжение на выходе транзистора, которое не изменяется с ростом тока базы. Однако…

За тот короткий промежуток времени, что показан на рис. 17.16 мощность рассеяния может существенно превышать наше предыдущее рассмотрение. Вот как выглядит импульс при увеличении частоты генератора до 1 МГц.

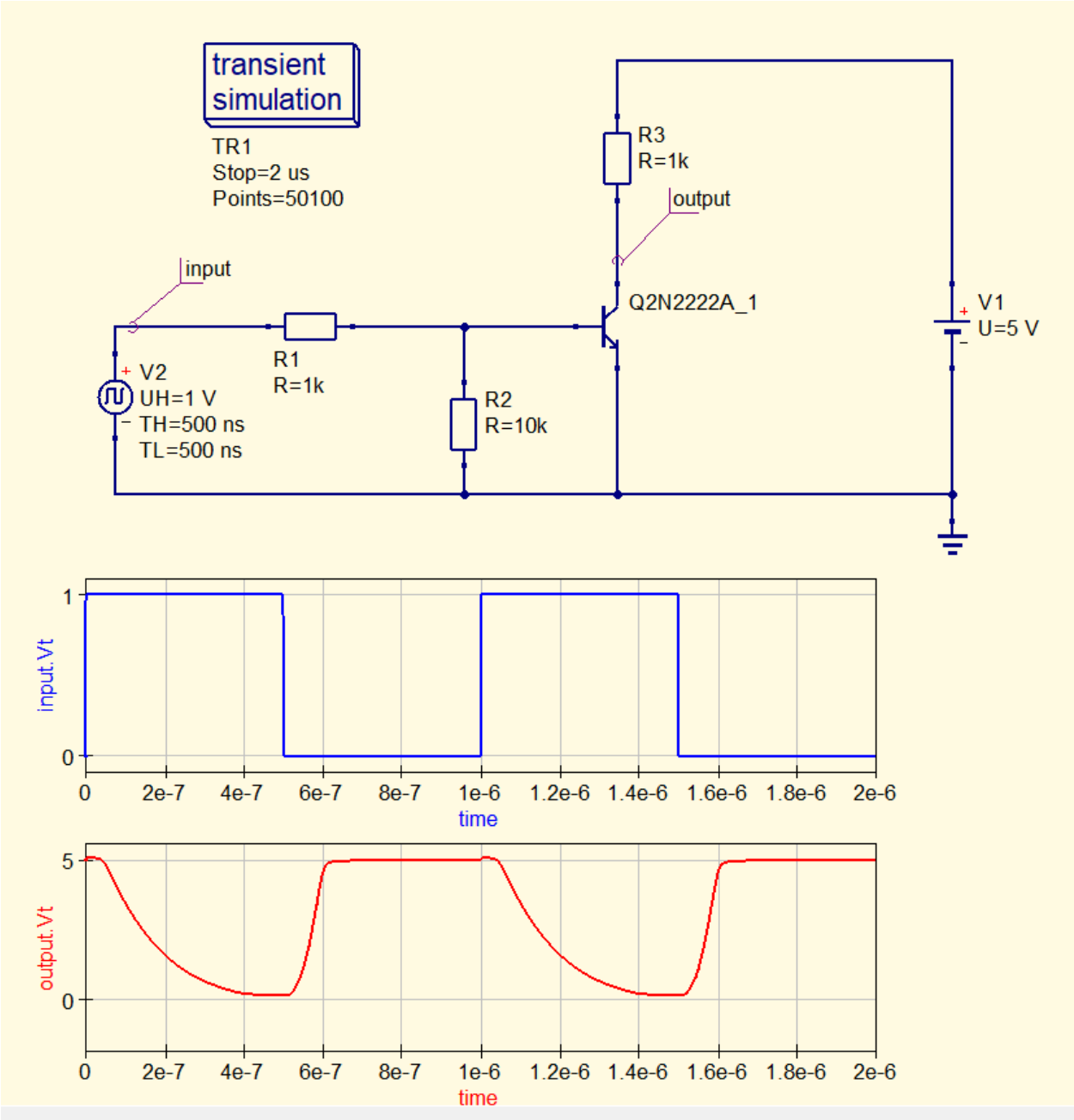


Рис. 17.17. Импульсы на выходе и выходе транзистора при частоте 1 МГц

И есть ещё один момент, который вызвал у меня удивление, и с чем нужно разбираться.

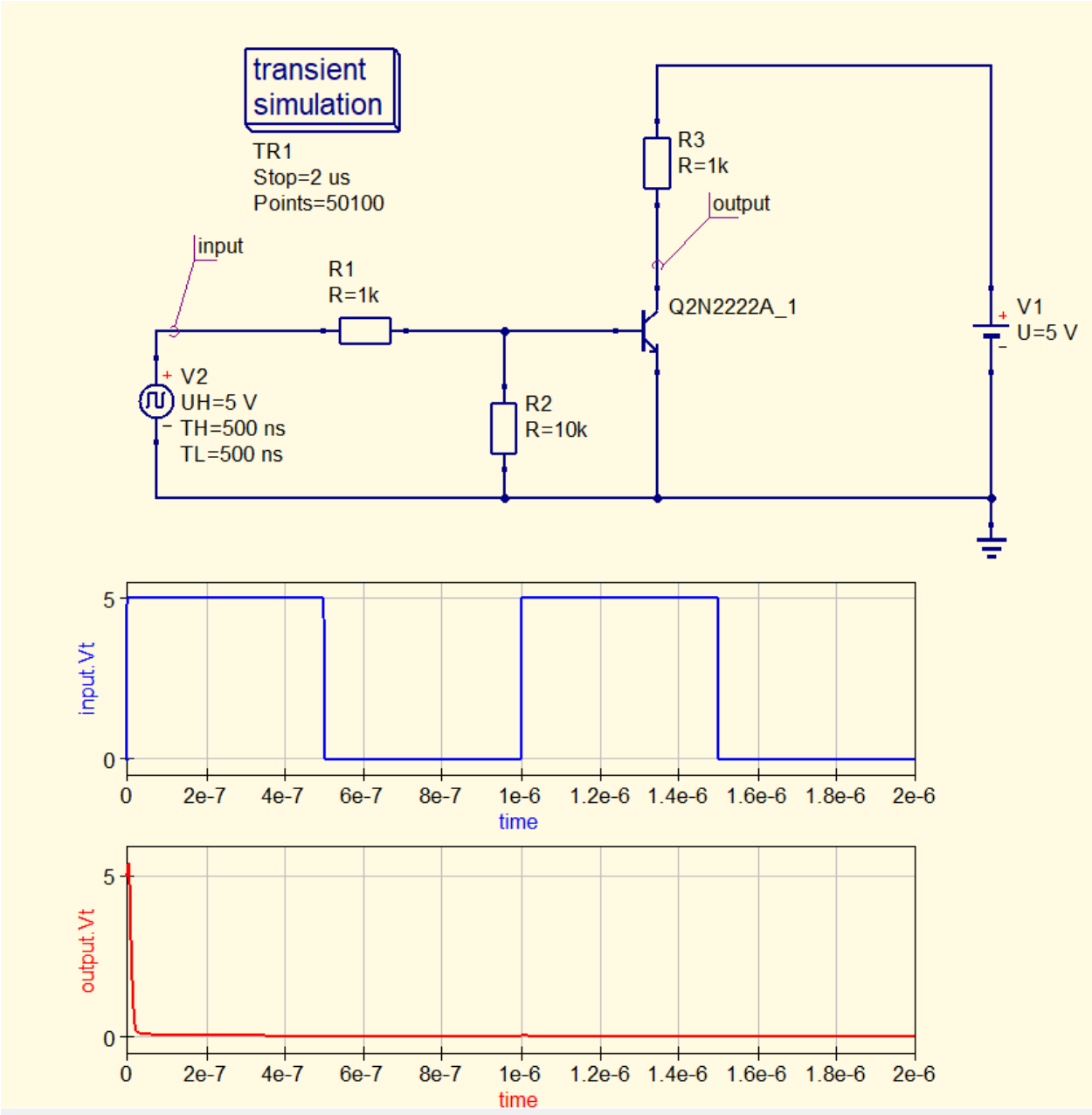


Рис. 17.18. Моделирование сигнала с амплитудой 5 В на входе

С разборкой полётов, впрочем, повременим. Могу пока только сказать, что программа моделирования QucsStudio может учитывать реальные параметры транзисторов. А для транзистора 2N2222A есть такой параметр, который называется временем выключения, которое составляет (можете проверить в справочнике) около 300 нС, но при токе коллектора 150 мА и напряжении на коллекторе порядка 30 В, что не отвечает нашему случаю. И поведение транзистора при иных исходных данных требует отдельного рассмотрения. Мне только хотелось сказать, что работа транзистора в ключевом режиме требует самостоятельного рассмотрения в силу имеющихся особенностей этого режима работы.

Завершить эту главу я хочу словами известной песни: «И пока я песню пела, пять минут уж пролетели». Пока я рассказывал о работе программ моделирования, пришло обновление. Правда, мой компьютер, похоже, не сможет поддерживать полновесную версию Windows 11, о чём я получил предупреждение и совет по выходу основной версии Windows 11 произвести чистую установку Windows 10, но посмотрим. Пока это выглядит так:

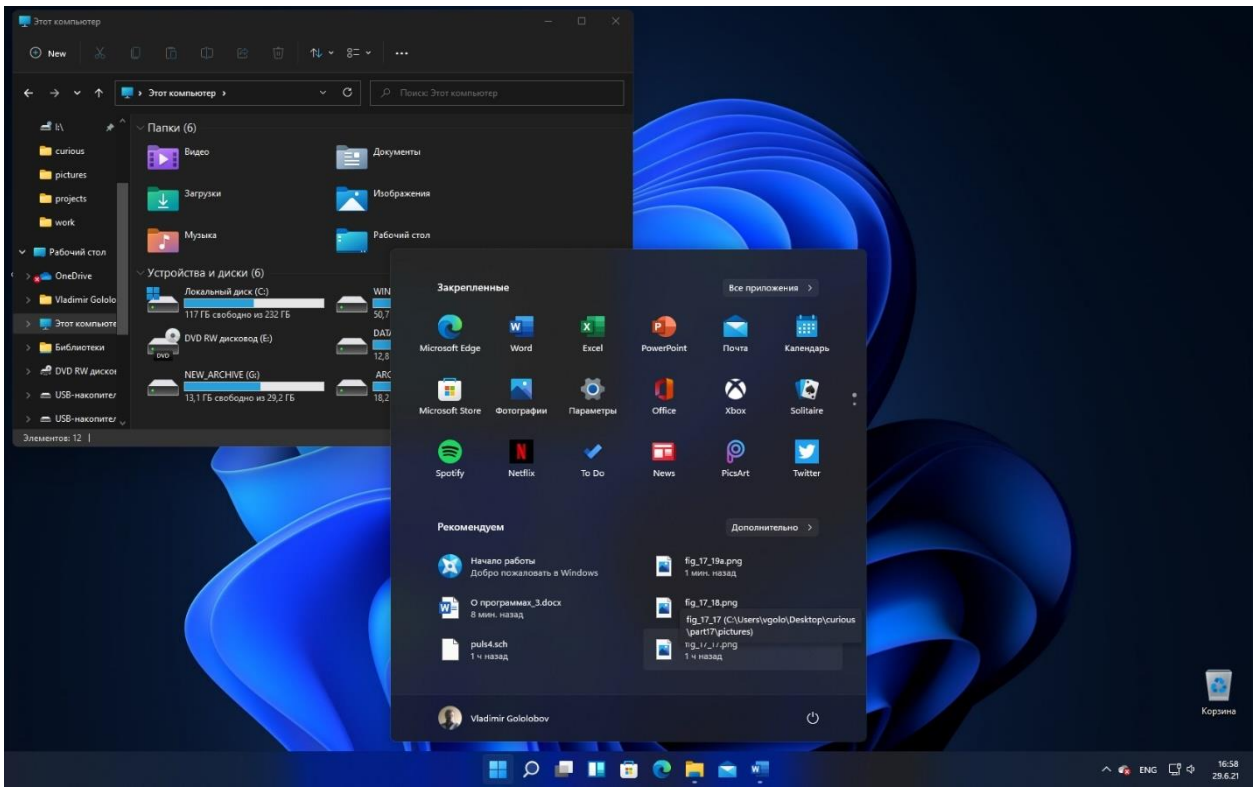


Рис. 17.19. Обновление до Windows 11

Глава 18. ElementaryOS и MPLABX

В прошлый раз при попытке установить MPLABX я столкнулся с тем, что сделал виртуальный жёсткий диск малой вместимости, отчего при установке среды разработки программ для PIC-контроллеров операционная система была испорчена, перестала работать. В этот раз я сделал диск более вместительным, что позволило установить и MPLABX, и компилятор XC8. Для начинающих любителей, я полагаю, этого более чем достаточно.

Моё знакомство с программой MPLAB произошло давно. В тот момент я выбирал микроконтроллер для рассказа о том, как работать с этим универсальным устройством. Мой выбор пал на PIC16F628A по причине как доступности контроллера при покупке, так и его цены. Дополнительным аргументом стало наличие перевода описания (datasheet) к этому микроконтроллеру. Начинающим радиолюбителям, как мне казалось, удобнее иметь описание на родном языке.

Сочинять что-то мне было не просто – порой приходилось бросать всё на полуслове, чтобы с головой уйти в основную работу. По возвращении уже было не вспомнить, что я хотел написать. В итоге, появление ошибок, одну из которых я заметил позже, а «гуру», когда книга вышла в бумажной версии, написали, что книга скорее вредная, чем полезная. Я был бы признателен опытным специалистам, если бы о моих ошибках они сообщили мне до выхода книги, но… Вот как это было…

Умный дом. Виртуальная лаборатория (2005 г)

Основы работы в среде MPLAB

После загрузки программы появляется рабочее окно.

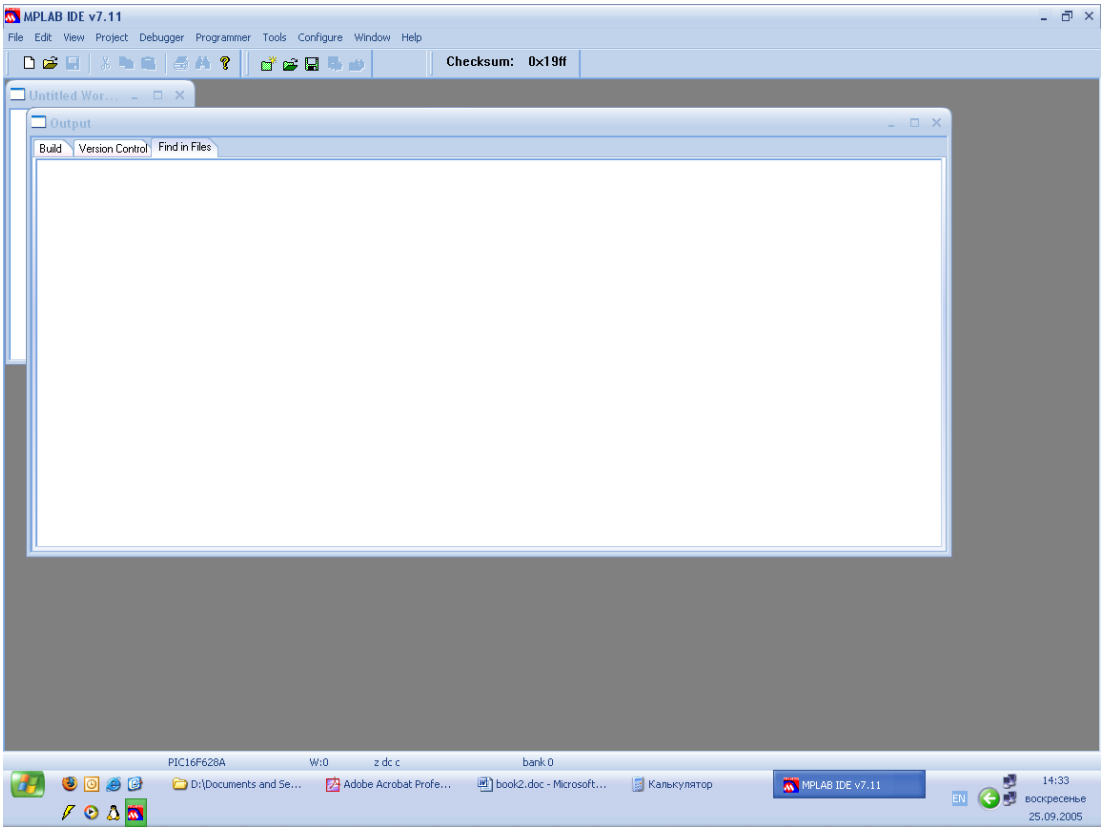


Рис. 18.1. Рабочее окно программы MPLAB v.7.11

Вид программы обычен для Windows, и, думаю, не требует особых пояснений.

Первое, что мы сделаем, создадим новый проект, в основном меню Project-New. Задаем название проекту relay, в папке, например, Relay, которую я советую открыть в основном разделе диска в папке MPLAB. Неоднократно я сталкивался с проблемой, которая не всегда очевидна. Многие программы, да это и удобно, предлагают хранить проект в папке «Мои документы». Проблема не возникает, если вы пользуетесь англоязычной версией Windows или русскоязычной версией программы. Но многие специализированные англоязычные программы начинают творить чудеса, если вы работаете в русскоязычной версии операционной системы.

Впервые я столкнулся с этим, когда одна из сред программирования при компиляции программы стала выдавать ошибку в строке -1. Что она имела в виду под строкой с отрицательным номером, я не знаю. Но отыскать ошибку в правильно написанной программе оказалось не так просто. Ошибка крылась в том, что программа, предлагая через операционную систему сохранить проект в папке «Мои документы», эту папку распознать не могла.

После создания нового проекта появляется окно навигатор проекта. Теперь выберем микросхему контроллера в разделе основного меню Configure-Select Device. Выбираем PIC16F628A.

Выглядит это, примерно, так:

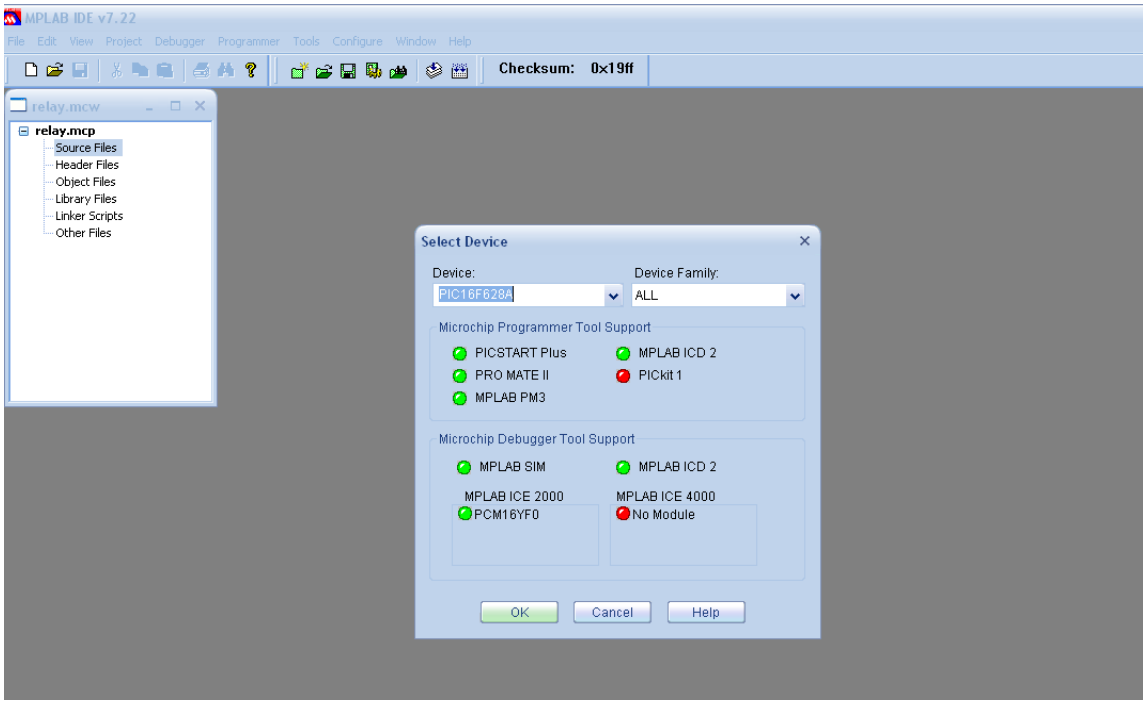


Рис. 18.2. Выбор конкретной модели микроконтроллера и программатора

Завершив выбор, следует создать файл основной программы. Выбираем File-New. Появляется окно редактора. Сохраним файл под именем relay.asm (File-Save As).

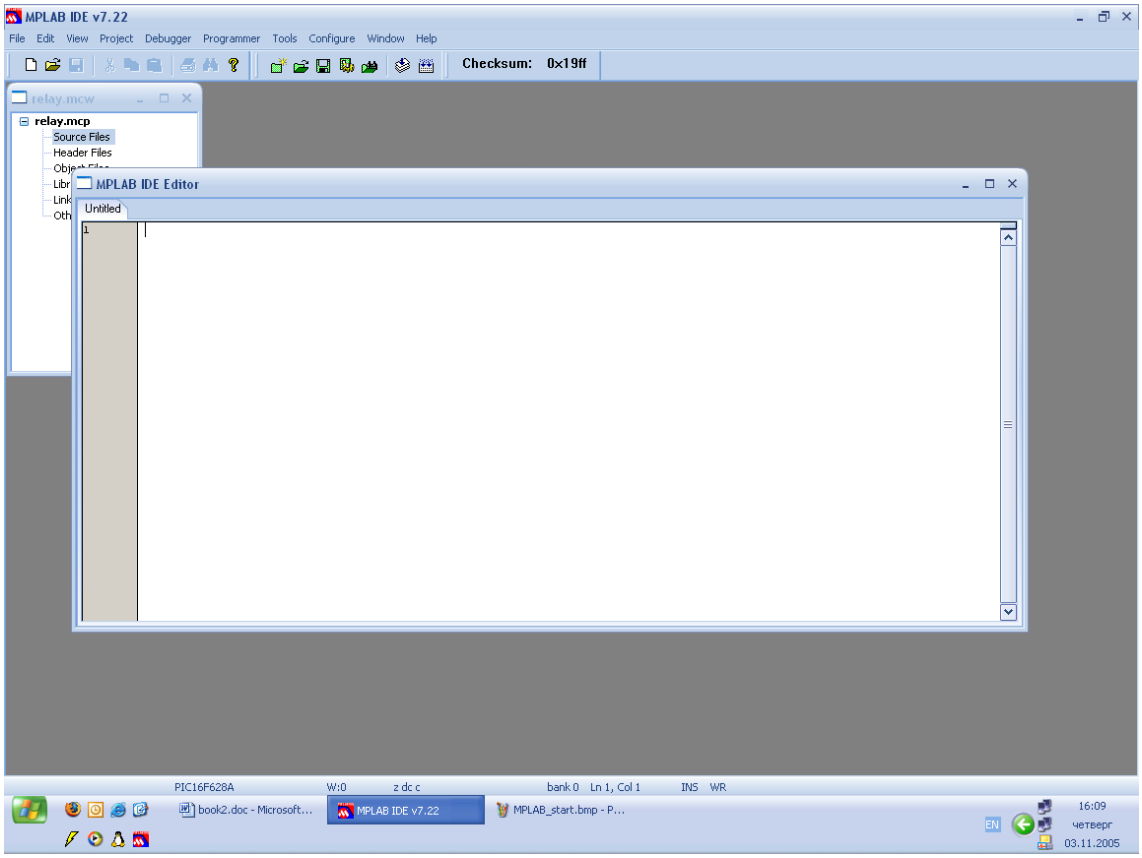


Рис. 18.3. Окно редактора для ввода текста программы

Далее добавим этот файл в проект. Для этого щелкнем правой кнопкой мыши по разделу Source Files в окне навигатора. В открывшемся меню выбираем Add Files, и указываем свой файл.

Для начала перенесем в окно редактора (прямым копированием из текста), небольшой фрагмент программы: инициализация, плюс сама программа, плюс две подпрограммы CALL adrsim и CALL cmnd. Последние в усеченном виде. Адрес будем использовать в данный момент «01». Не забудем поставить END в конце программы.

```
adrsim:     CLRW        ; Если адрес 1 запишем символы «0» «1» (30h и 31h)
            ADDLW 0x30
            MOVWF 0x21
            CLRW
            ADDLW 0x31
            MOVWF 0x22
            MOVF 0x20, 0
            BCF STATUS, Z
            XORLW 0x1
            BTFSC STATUS, Z
            RETURN

cmnd:       BCF STATUS, Z
            MOVF RCREG, 0
            XORLW 0x52          ; Проверим наш ли модуль R (52h)
            BTFSS STATUS, Z     ; Если нет, вернемся
            RETURN
in1:        BTFSS PIR1, RCIF  ; Ждем прихода первого символа адреса
            GOTO in1          ; Если совпадает, продолжим
            MOVF RCREG, 0
```

```
            BCF STATUS, Z
            XORWF 0x21, 0 ; Первый символ адреса, запомненный в регистре 21h.
            BTFSS STATUS, Z
            RETURN

in2:        BTFSS PIR1, RCIF  ; Ждем прихода второго символа адреса
            GOTO in2          ; Если совпадает, продолжим
            MOVF RCREG, 0
            BCF STATUS, Z
            XORWF 0x22, 0 ; Второй символ адреса, запомненный в регистре 22h.
            BTFSS STATUS, Z
            RETURN
```

Для отладки откроем окно наблюдения View-Watch, в котором выберем регистры STATUS, WREG, PIR1, EEDATA, RCREG, 20h, 21h, 22h, 30h. Необходимые регистры открываются кнопкой с обозначением стрелки вниз, правее названия регистра, которое, в свою очередь, рядом с кнопочкой ADD SFR. Ее следует нажать после выбора регистра.

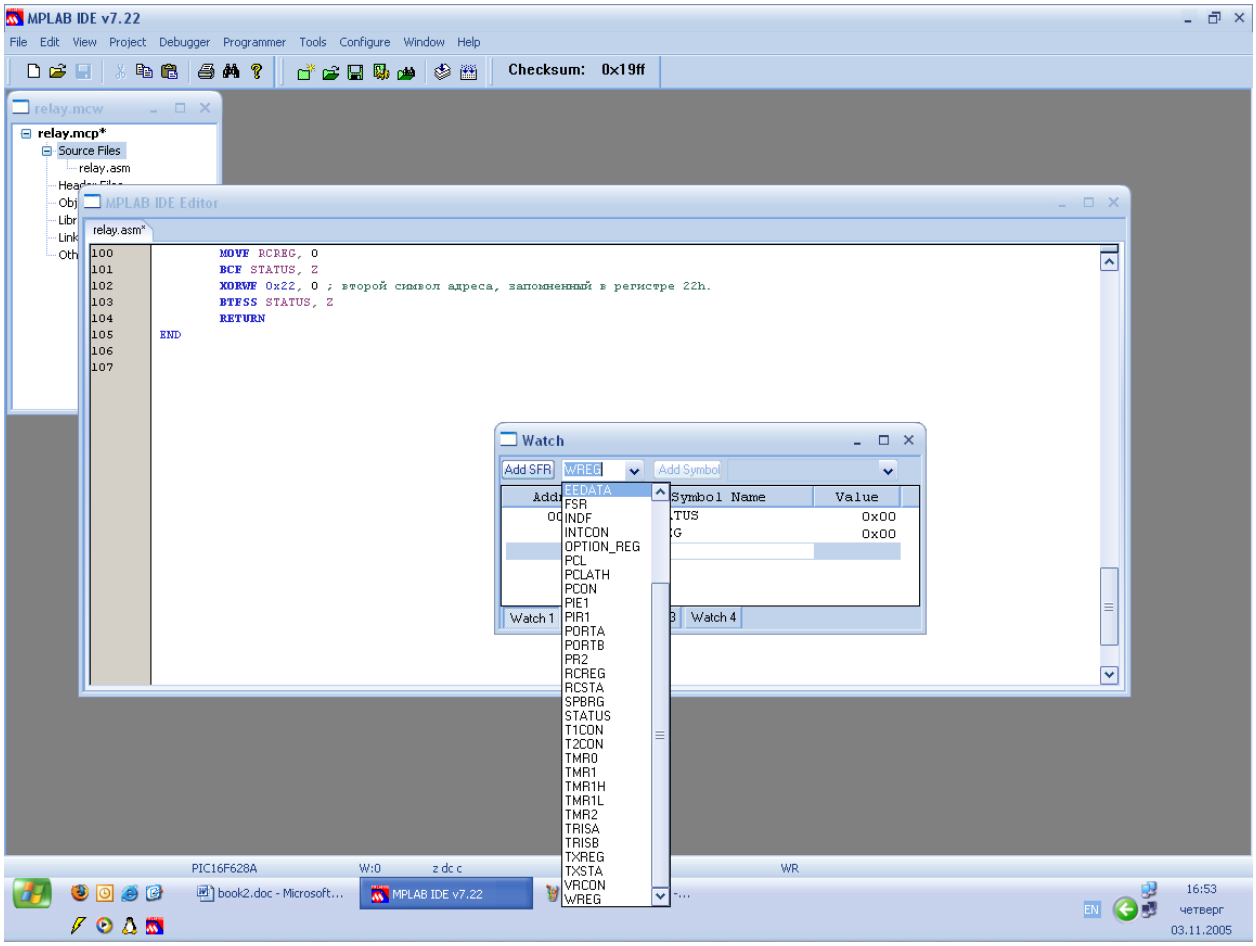


Рис. 18.4. Выбор регистров для наблюдения

Регистры без имени (20h, 21h и т.д.) мы добавляем, просто вводя адрес в колонку Address на новой строке.

Добавим и EEPROM через выбор View - EEPROM. Установим в качестве симулятора программный (Debugger-Select Tool-MPLAB SIM). Создадим два файла input.txt и output.txt (File-New). В файле input.txt, который открывается в окне редактора, запишем слово команды “R01$” в виде строки в

кавычках. Сохраним этот файл (File-Save). Теперь сделаем установки отладчика (Debugger-Settings…): частоту процессора зададим 4 мГц, на вкладке Uart1 IO установим опцию Enable Uart1 IO, укажем входной файл input.txt (browse) и выходной output.txt. Подтвердим замену последнего файла, и установим опцию Rewind Input. Изменим значение на вкладке Animation/Realtime на 1 мс. Нажмем «Применить» и «ОК». Включим в наши окна View-Output. Создадим новый сценарий, который позволит нам ввести адрес, имитируя переключатель. Для этого выберем в основном меню Debugger-Stimulus Controller-New Scenario. Выберем нужный нам RB4 в окне Pin/SFR. В окне Action выберем Set High. Нажмем кнопочку со стрелкой вправо рядом с этим в колонке Fire. Сохраним сценарий под именем relay и свернем его (не покинем, а свернем!). Теперь откроем окно (Configure-Configuration Bits), где установим биты, записываемые по адресу 2007h. Слово должно получиться 3F1Ah. Закроем это окно.

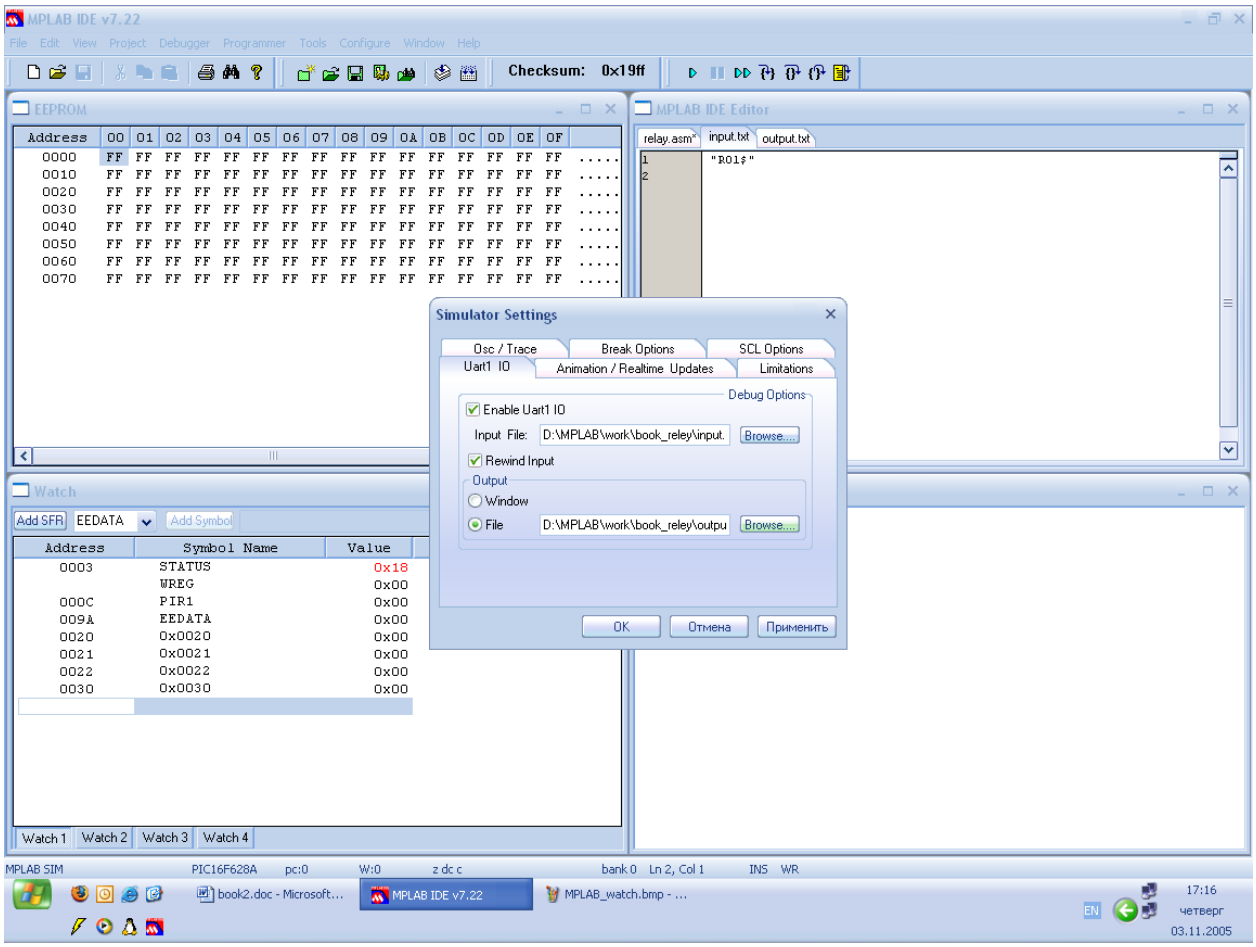


Рис. 18.5. Задание установок отладчика

В завершение упорядочим окна (Windows-Tile Horizontally), сохраним все (File-Save All), и добавим в проект файл todo.txt, который предварительно создадим, а в менеджере проекта поместим его в раздел Other Files. В файле todo.txt будем вести план работы.

Теперь откомпилируем проект Project-Build All. Сохраним и вид проекта File–Save Workspace. Мы готовы к отладке программы.

После выхода из программы MPLAB, и подтверждения сохранения вида проекта, новой загрузки программы и открытия проекта (Project-Open) мне приходится обнулять адрес 0h в EEPROM, нажимать Fire в Scenario, и вводить адреса 20h, 21h и т.д. в окне Watch, которые программа пишет Not Found. Я не уверен, что дернул за все веревочки, но…

Если вы правильно перенесли программу, то, нажав на инструментальной панели кнопочку обозначенную ►► вы увидите анимацию, а в регистре RCREG (приемный регистр USART) появятся шестнадцатеричные коды символов из строки файла input.txt. Можно вписать в ячейку по адресу 0h EEPROM значение 1h, и увидеть, как она переписывается в регистр 30h.

Есть еще несколько полезных возможностей. Одна из них проверить работу с некоторого места. Для этого остановим анимацию (кнопочкой ▌▌), установим курсор к нужной строке и нажмем правую клавишу мышки. Выберем в выпадающем меню Set PC at Cursor. Теперь, нажимая значок Step Into{¬}, мы можем отследить все изменения.

В окончательном виде я работаю в среде, которая выглядит так:

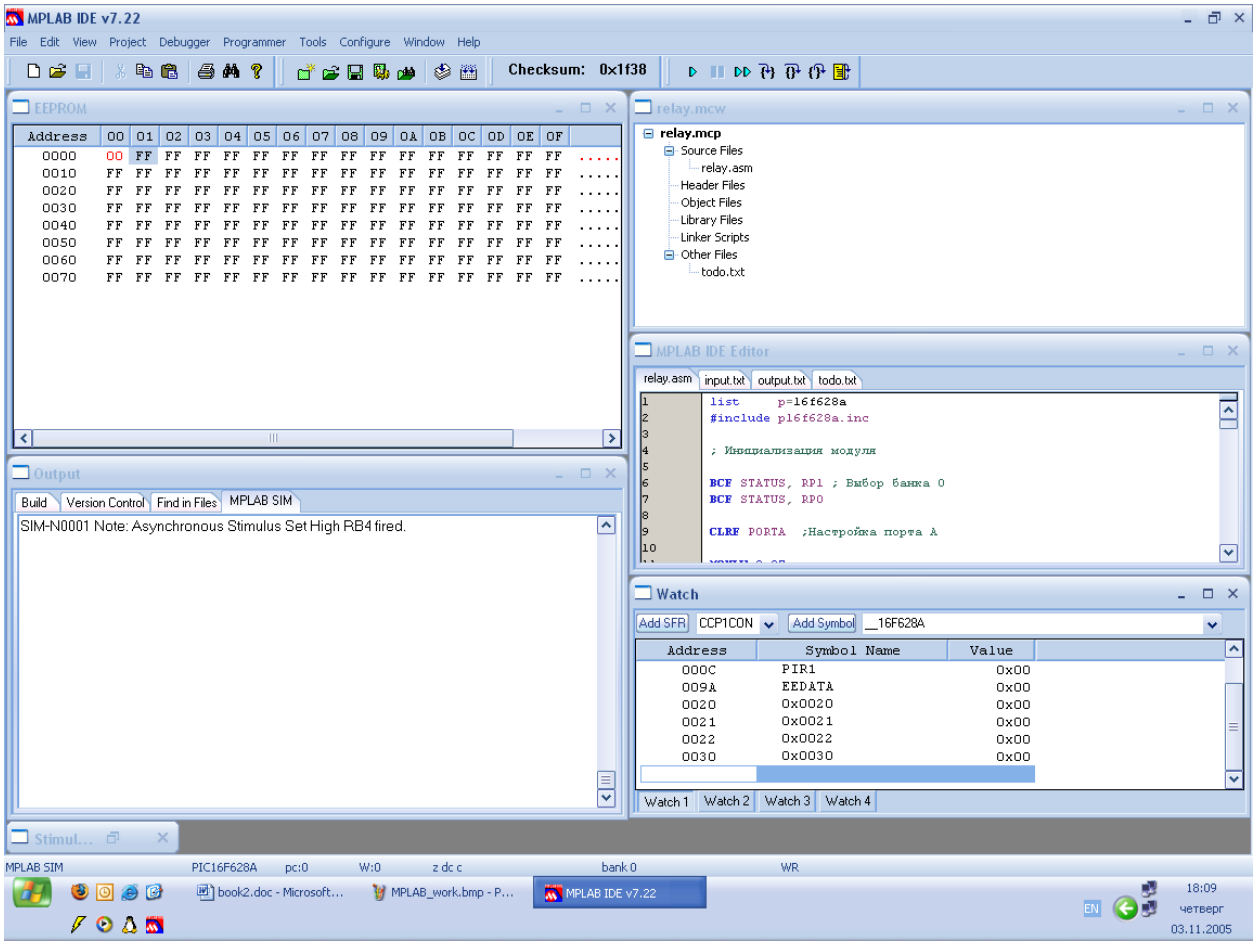


Рис. 18.6. Выбор удобной среды для работы

Кроме написания программы на ассемблере, среда MPLAB поддерживает написание программ на языке «С». Существуют компиляторы разных производителей. Я использую демонстрационную версию кросс-компилятора Hi-Tech. Посмотрим, не будет ли проще написать предыдущую программу на языке «С»?

Релейный модуль, версия программы на языке «С»

Выбор языка программирования происходит при задании в Project-Select Language Toolsuite:

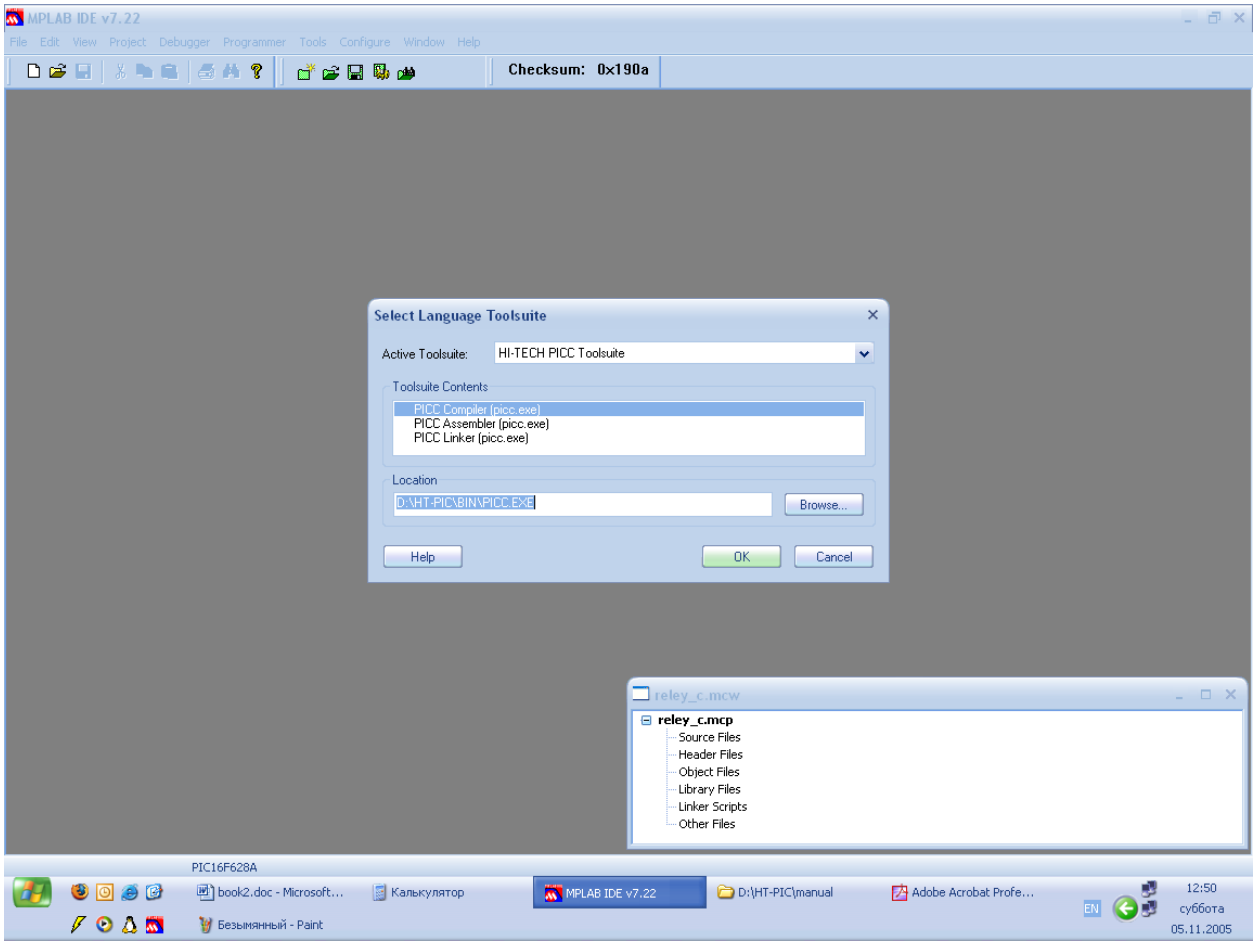


Рис. 18.7. Выбор компилятора

Создадим файл заголовка и основной файл. Добавим в файлы заголовка нужный нам контроллер. Остальная часть работы мало, чем отличается от работы на ассемблере. В программе я не сохраняю состояние реле в EEPROM. В окончательном виде файл заголовка:

```
#define MODULNAMESIM 'R'
#define CMDSIM '$'
#define bitset(var,bitno) ((var) |= 1 << (bitno))
#define bitclr(var,bitno) ((var) &= ~(1 << (bitno)))

void putch(unsigned char);
unsigned char getch(void);
int  init_comms();
int sim_num_adr();
int cmd();
int rel_on(int num);
int rel_off(int num);
int rel_stat(int num);
```

Основной файл:

```
#include <pic16f62xa.h>
#include <stdio.h>
#include "releу_c.h"
```

```
unsigned char input;        // Считываем содержимое приемного регистра
unsigned char MOD_SIM1;     // Первый символ адреса модуля
unsigned char MOD_SIM2;     // Второй символ адреса модуля
unsigned char REL_SIM;      // Символ реле
unsigned char COMMAND;      // Символ команды
int MOD_ADDR;               // Заданный адрес модуля, как число
int sim1_num = 0;
int sim2_num = 0;
int sim_end_num = 0;
int MOD_NUM;                // Полученный адрес модуля, как число
int REL_NUM;                // Номер реле
int RELSTAT = 0;            // Статус реле (позиционно): 1 - вкл, 0 -
выкл.
```

/* Получение байта */

```
unsigned char getch()
{
      while(!RCIF)          // Устанавливается, когда регистр не пуст
            continue;
      return RCREG;
}
```

/* вывод одного байта */

```
void putch(unsigned char byte)
{
      PORTB = 1;            // Переключим драйвер RS485 на передачу
      TXEN = 1;             // Разрешаем передачу
      while(!TXIF)          // Устанавливается, когда регистр пуст
            continue;
      TXREG = byte;
}
```

/* Преобразуем символьный адрес в число*/

```
int sim_num_adr()
{
      sim_end_num = 0;
      sim1_num = getch();         // Чтение первого символа номера
      MOD_SIM1 = sim1_num;        // Сохраним первый символ
      sim2_num = getch();         // Чтение второго символа номера
      MOD_SIM2 = sim2_num;        // Сохраним второй символ
      sim1_num = sim1_num - 0x30;      // От первого символа к числу
      sim2_num = sim2_num - 0x30;      // От второго символа к числу
      sim_end_num = sim1_num*0x0A + sim2_num; // Объединим в одно число
      return sim_end_num;
}
```

/* Получение и выполнение команды */

```
int cmd()
{
input = getch();
      REL_NUM = input;                  // Номер реле в символьном виде
      REL_SIM = REL_NUM;
      REL_NUM = REL_NUM - 0x30;         // Номер реле в числовом виде


switch (COMMAND = getch())              // Прочитаем команду
{
      case 'N': rel_on(REL_NUM);        // Если команда включить
      break;
      case 'F': rel_off(REL_NUM);       // Если команда выключить
      break;
      case 'S': rel_stat(REL_NUM);      // Если команда передать состояние
      break;
      }
}
```

/* Выполнение команды включения заданного реле */

```
int rel_on(int num)
{
      bitset (RELSTAT, REL_NUM);     // В переменной RELSTAT побитовое
состояние реле
      PORTA = RELSTAT;       // Установив бит, переписываем в порт A
}
```

/* Выполнение команды выключения заданного реле */

```
int rel_off(int num)
{
      bitclr (RELSTAT, REL_NUM);
      PORTA = RELSTAT;            // Сбросив бит, переписываем в порт A
}
```

/* Выполнение команды передачи состояния заданного реле */

```
int rel_stat(int num)
{
      putch('R');            // Отправляем символ R
      putch(MOD_SIM1);       // Первый символ номера модуля
      putch(MOD_SIM2);       // Второй символ номера модуля
      putch('#');            // Символ статуса
      putch(REL_SIM);        // Символ номера реле
      if ((RELSTAT>>REL_NUM)&0x01) putch('N');  // Проверяем, установлен
ли бит?
      if (!((RELSTAT>>REL_NUM)&0x01)) putch('F'); // Проверяем, сброшен
ли бит?
            putch(0x0A); // Только для вывода в файл!!!!!!
}
```

```
int init_comms()      // Инициализация модуля
{
PORTA = 0x0;          //  Настройка портов A и B
CMCON = 0x7;
TRISA = 0x80;
TRISB = 0xF6;

RCSTA = 0x90;         //  Настройка приемника
TXSTA = 0x4;          //  Настройка передатчика
SPBRG = 0x16;         //  Настройка режима приема-передачи

INTCON=0;             //  Запретить прерывания
PORTB = 0;            //  Выключим передатчик драйвера RS485
}


void main(void)

{
```

/* Инициализация модуля */

```
init_comms();
```

/* Прочитаем и преобразуем номер модуля */

```
      MOD_ADDR = PORTB;           // Номер модуля в старших битах
      MOD_ADDR=MOD_ADDR>>4;       // Сдвинем на четыре бита
```

/* Начинаем работать */

```
start:     input = getch();
      while(input != MODULNAMESIM) input = getch();    // Ждем, если не
наш модуль
      MOD_NUM = sim_num_adr();         // Чтение из сети (файла)
      if (MOD_NUM == MOD_ADDR)         // Если наш адрес модуля
      {
            input = getch();
            if (input == CMDSIM) cmd(); // Если символ команды
      }
            goto start;
}
```

Это не шедевр программы на языке «С», но, вроде бы, работает. В качестве входного файла команд я использовал input.txt (примечания, которые я сделал, только для книжного варианта, в файле их делать не следует) такого вида:

"noperat" *Проверим, не будет ли мешать посторонняя команда*
"L03$3N" *Проверим, не отвечает ли модуль на обращение к другим модулям*
"R11$2N" *Проверим, не отвечает ли модуль на чужие адреса*
"R03$2N" *Включим второе реле третьего релейного модуля*

"R01$2S" *Проверим, не отвечает ли модуль на чужие адреса*
"R03$2S" *Запросим состояние второго реле третьего релейного модуля*
"R15$1N" *Проверим, не отвечает ли модуль на чужие адреса*
"R03$1N" *Включим первое реле третьего релейного модуля*
"R03$1S" *Запросим состояние первого реле третьего релейного модуля*
"R03#1F" *Проверим, не отвечает ли модуль на передачу состояния*
"R03$1F" *Выключим первое реле третьего релейного модуля*
"R03$1S" *Запросим состояние первого реле третьего релейного модуля*
"R03$2S" *Запросим состояние второго реле третьего релейного модуля*

Получаем выходной файл output.txt:

R03#2N

R03#1N

R03#1F

R03#2N

Теперь, пока вы опробуете работу в среде MPLAB, я, пожалуй, прерву работу над книгой. Поеду в магазин «Чип и Дип», что на Земляном валу, возле Курского вокзала, куплю все необходимое для сборки конвертора RS232-RS485, программатора и создания прототипа. Затем соберу прототип на макетной плате, а когда закончу и проверю, поделюсь впечатлениями...

Микроконтроллер глазами начинающего (2012 г)

В Москве, думаю и в других городах, в последние годы появились новые светофоры на регулируемых перекрёстках, оснащённые «светофорами» для пешеходов. На специальном табло высвечивается время, оставшееся до переключения светофора, на втором табло красным высвечивается пиктограмма стоящего пешехода; когда переход открывается для пешеходов, появляется зелёная пиктограмма движущегося пешехода и (тоже зелёным) время, оставшееся до закрытия перехода для пешеходов.

К чему это?

Я предлагаю начать освоение микроконтроллера с создания модели такого светофора. Если вы задумаете довести работу до конца, то подобную модель можно использовать в школе на уроках, посвящённых правилам дорожного движения. Если это вам не нужно, то: как среди чисел нет плохих и хороших, интересных и неинтересных, так среди программ нет смысла искать интересные программы. Для каждого из вас есть одна единственная, которая интересна вам (остальные неинтересны), но всё, о чём мы будем говорить, относится и к той единственной, что интересна именно вам. Разбирая «устройство программы», мы попутно коснёмся сопутствующих вопросов таких, как – зачем использовать транзисторные ключи, как работать с матричными конструкциями и т.д.

Перед тем, как перейти, собственно, к программе, я хочу поделиться своими впечатлениями о создании рабочей среды: об установке программы MPLABX. Я установил эту программу и загрузил, и установил оба компилятора в операционной системе Windows Vista (для SDCC, кажется, понадобится установка пакета gputils). Думаю, что установка без проблем осуществима и в Windows XP, и в Windows 7. Однако при установке программы в Fedora 17 я столкнулся с тем, что

потребовалось удалить пакет java версии 7 (что удалило пакет LibreOffice), установить пакет jdk версии 6, найденный на сайте Oracle. Удобно ввести в поисковую строку браузера запрос java se download, который приведёт на сайт oracle.com:

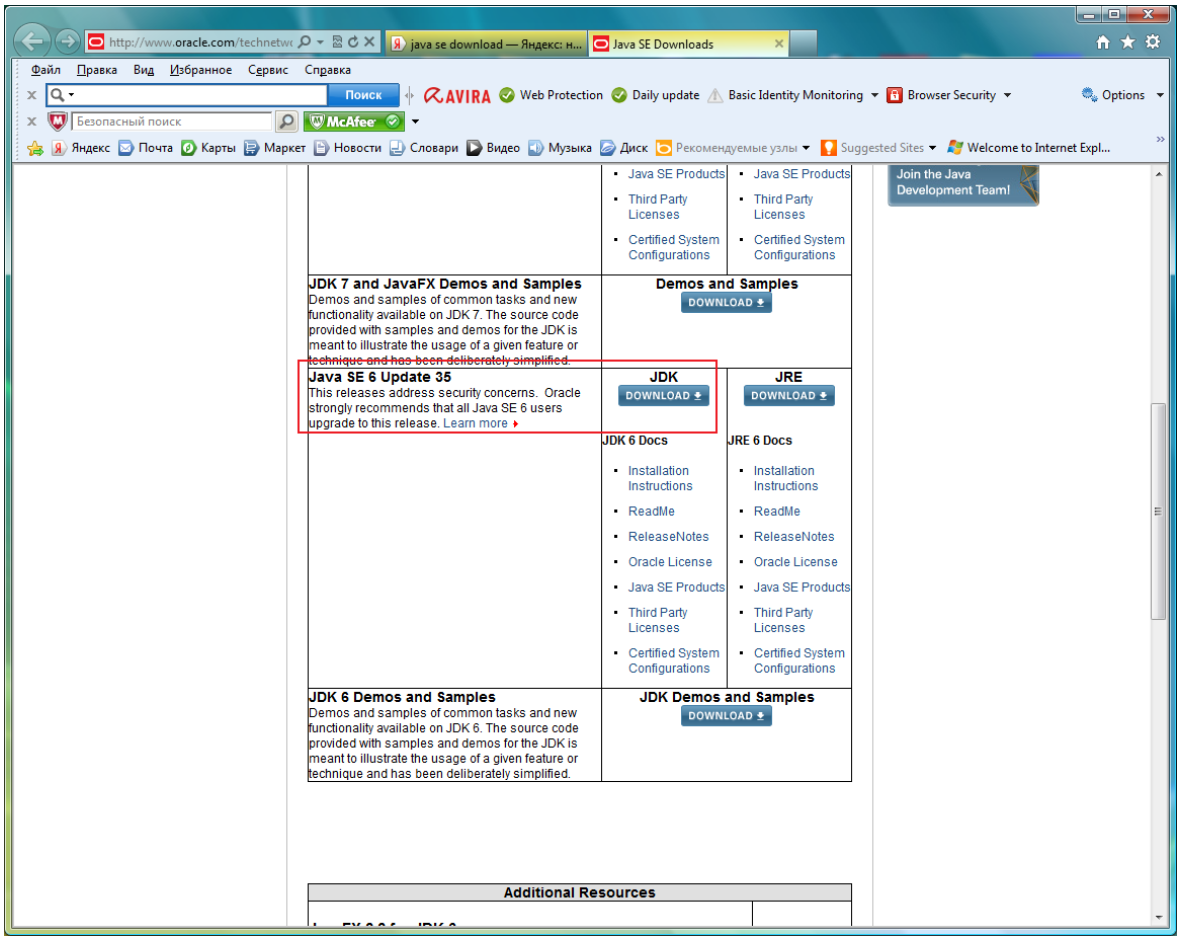


Рис. 18.8. Сайт загрузки java 6

К загрузке предлагается два пакета для Linux: jdk-6u35-linux-i586.bin и jdk-6u35-linux-i586-rpm.bin. Установка требует только согласия (поставить галочку в окне соглашения) с правилами использования программы. Первый пакет распаковывается в домашнюю папку как пакет java, второй даёт набор пакетов для установки rpm. Надеюсь, вы будете последовательны в своих действиях (в отличие от меня) и быстро выясните, что и как следует устанавливать.

Для запуска установки следует, возможно, предварительно в свойствах этих файлов указать, что они исполняемые (щелчок правой кнопкой мышки по файлу и выбор из выпадающего меню свойств), и использовать терминал. Вначале командой cd /путь_к_файлу, например, я устанавливал из домашней папки: cd /home/Vladimir, перейти в место, где находится пакет, а затем использовать команду ./jdk-6u35… и т.д., то есть имя файла. В Fedora 17 мне не потребовалось устанавливать пакет с компилятором SDCC, что пришлось сделать в ALTLinux и openSUSE.

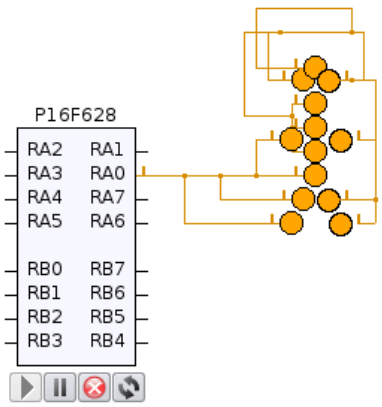
Компилятор устанавливается аналогично, а после распаковки файла с компилятором, можно обнаружить инструкцию по установке SDCC, которая, думаю, поможет вам без особенных хлопот установить и пакет java. Добавления могут коснуться переноса (с правами root) всех файлов в файловую систему (из домашней папки, например) и задания путей в переменных окружения.

Советы, как установить java 6, можно поискать в Интернете. После установки компилятора SDCC и java 6 можно установить MPLABX. С правами root (опять проверьте, файл MPLABX должен быть исполняемым) в файловом менеджере достаточно запустить установку двойным щелчком, как в Windows.

В репозитории ALTLinux (и для openSUSE я нашёл пакет установки) есть компилятор sdcc, но попытка использовать его привела в обоих дистрибутивах к ошибкам, что и заставило меня скачать последнюю версию компилятора с сайта производителя. После замены версии компилятора ошибки исчезли, появилась возможность использовать и этот компилятор вдобавок к другим.

Но вернёмся к рассказу. Отчего-то я думаю, что проблемы использования MPLABX в Linux волнуют далеко не всех. Итак.

Я предпочитаю начинать рассказ с простых решений. В смысле решений, которые понятны даже непосвящённому. Предположим, что мы выложили фигурку пешехода из светодиодов. Как это выглядит мне удобнее показать в программе KTechlab:



Я не художник, что явно, но для наших целей это вполне подойдёт.

Можно было бы добавить светодиодов, но в этом случае легко запутаться с выводами.

Программу я, не сомневайтесь, предварительно написал и откомпилировал, иначе моделирование в KTechlab не получилось бы. Хотя вместо контроллера можно было бы использовать батарейку, но мы рассматриваем микроконтроллеры, и в таком виде всё выглядит…

Рис. 18.9. Фигурка стоящего пешехода из светодиодов

Формально для реализации программы достаточно выполнить команду RA0 = 1, записанную на языке Си. Но! Давайте разбираться с этими «но!»:

- Каждый из светодиодов потребляет, если мы возьмём обычные индикаторные светодиоды, ток около 15 мА.
- Вся фигурка потребляет ток более 150 мА.
- Порты микроконтроллера могут использоваться и для ввода данных, и для вывода данных, но микроконтроллеру нужно сообщить об этом!

Я привёл только несколько «но», с которыми начнём разбираться. Каждая модель микроконтроллера (это описано в datasheet, справке) может обеспечить только предельно допустимый ток на выходе, и это, как правило, не очень большой ток. Поэтому лучше между выводом микроконтроллера и фигуркой из светодиодов поставить промежуточный элемент – реле, а лучше транзистор с подходящим допустимым током коллектора.

Если вы посмотрите в справочник по диодам, то можете найти параметры для светодиода, которые могут выглядеть так: падение напряжения при токе 15 мА порядка 1.5 В. На выходе микроконтроллера напряжение, очень часто, порядка 5 В. Избыток напряжения обычно «гасят» с помощью резистора, который устанавливается последовательно с диодом.

Мы можем посчитать ток через все светодиоды (взяв напряжение на них 1.5 В) и по закону Ома и закону Кирхгофа определить величину добавочного резистора для всех светодиодов. Конечно, светодиоды элементы достаточно надёжные и работают без проблем долго, но, представим, что один из светодиодов вышел из строя. Ток через дополнительный резистор уменьшается, что увеличивает падение напряжения на светодиодах (за счёт уменьшения напряжения на резисторе). А это приводит к увеличению тока через светодиоды и, в свою очередь, может привести к выходу из строя другого (самого слабого в цепи) светодиода и т.д. Согласитесь, есть о чём задуматься.

Вернёмся к микроконтроллеру. Чтобы сказать ему, будет ли вывод работать на ввод информации или на вывод, используют специальные регистры, которые называются TRIS, то есть в нашем случае это будет регистр TRISA (для порта A). Будет ли вывод микроконтроллера работать на вход или на выход, определяется записью единицы в соответствующий бит в первом случае и ноля во втором. То есть, нам надо записать единицу в младший бит регистра TRISA. Это можно сделать так: TRISA = 0xFE. Запись 0x (ноль и икс) говорит компилятору, что мы используем шестнадцатеричное число. В двоичном виде мы получим ноль только в младшем бите.

Напишем нашу первую программу на языке Си. Загрузим MPLABX:

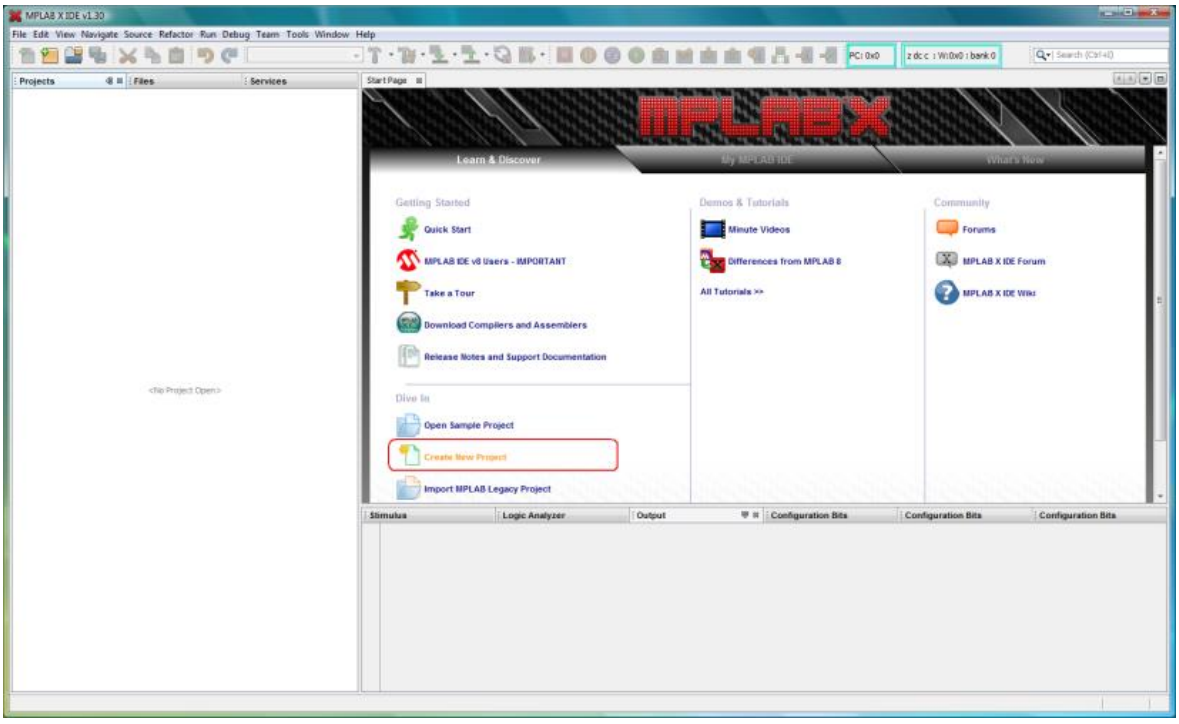


Рис. 18.10. Загруженная программа MPLABX

На рисунке я отметил раздел *Create New Project*. С этого раздела начинается создание любого нового проекта: щелчком по этому разделу левой клавишей мышки. Появляется диалоговое окно для выбора характера проекта:

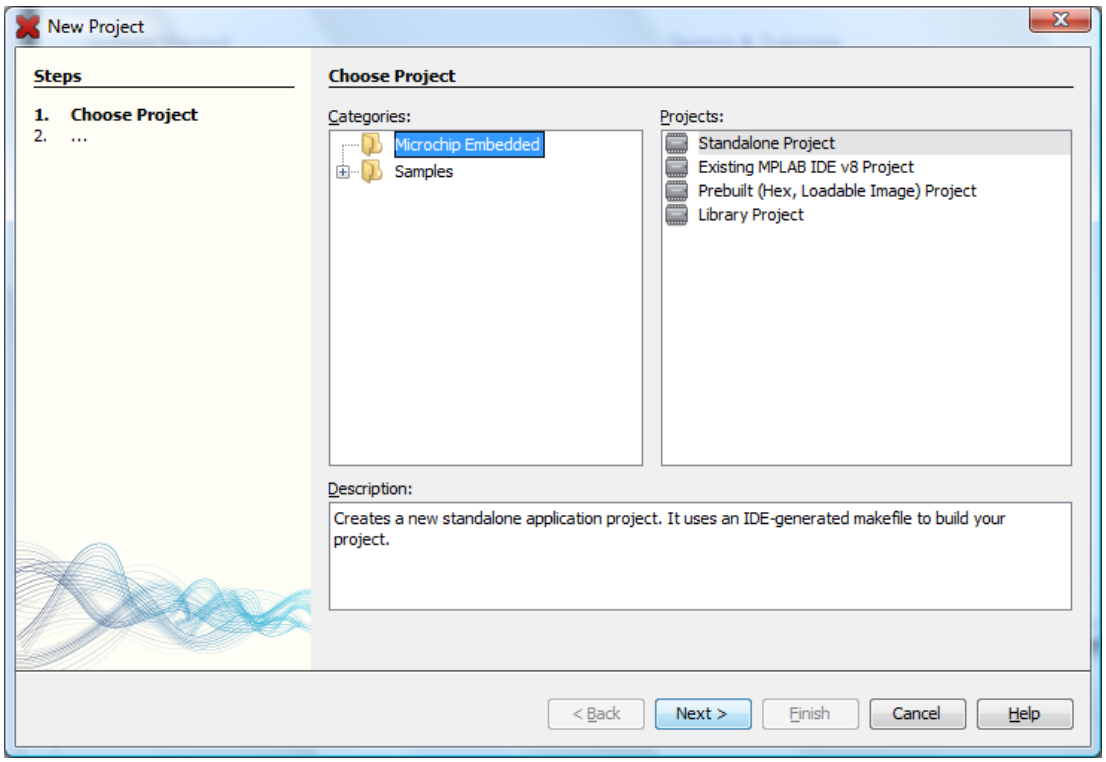


Рис. 18.11. Выбор характера проекта

Мы ещё не готовы к созданию чего-либо иного, чем *Standalone Project*, то есть, отдельного проекта. Поэтому нажимаем на кнопку **Next>** и продолжаем настройки проекта.

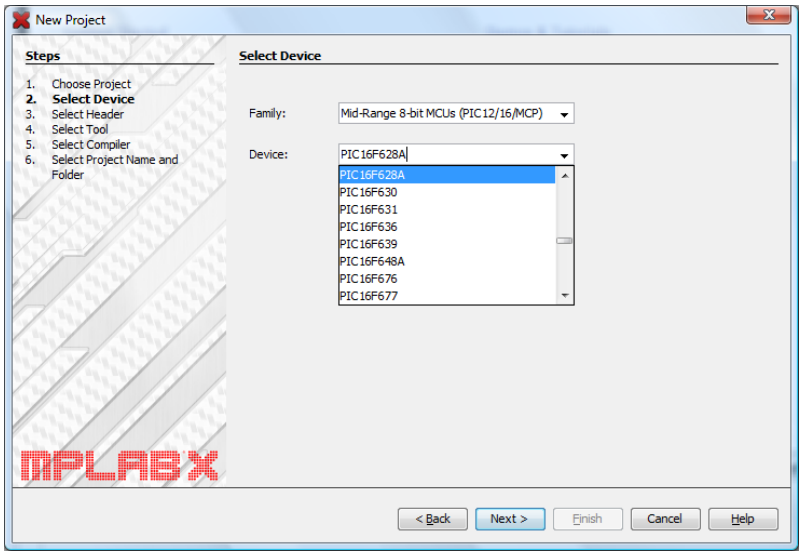


Рис. 18.12. Выбор модели микроконтроллера

Я уже говорил, что использую микроконтроллер PIC16F628A. Есть модели проще, есть модели с более привлекательными параметрами, но это довольно доступная модель, имеющая описание (datasheet) на русском языке, с этой моделью в виде PIC16F628 работает (как может) программа KTechlab и эту модель (под тем же именем) поддерживает компилятор SDCC. В общих задачах выбор модели зависит от ряда параметров, присущих именно самой задаче. Для всех, или почти всех, задач, которые мы попробуем решить, эта модель вполне подходит.

Если понадобится что-то исследовать, что не сможет обеспечить эта модель микроконтроллера, мы обратимся к другим моделям, но постараемся сделать это за компьютером. Нажимаем кнопку **Next>**. В следующем диалоге можно установить опцию использования заголовков для отладчика, но работать это будет в тех случаях, когда такая возможность обеспечена остальными параметрами проекта. Переход к следующему диалоговому окну позволяет выбрать программатор и отладочный аппаратный модуль:

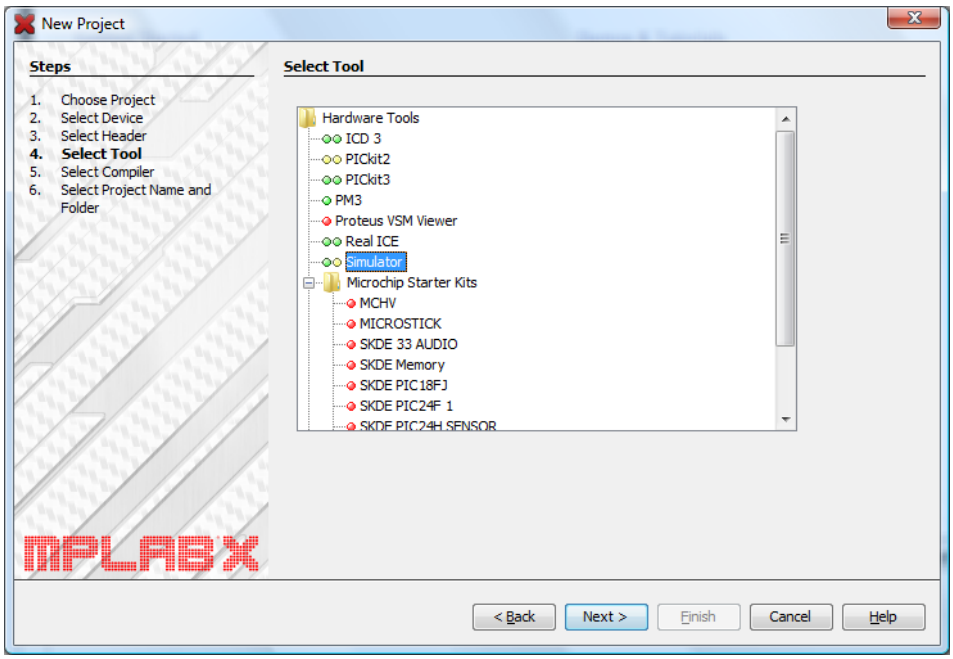


Рис. 18.13. Выбор отладчика и/или программатора

И, наконец, мы можем выбрать компилятор, с которым предпочитаем работать:

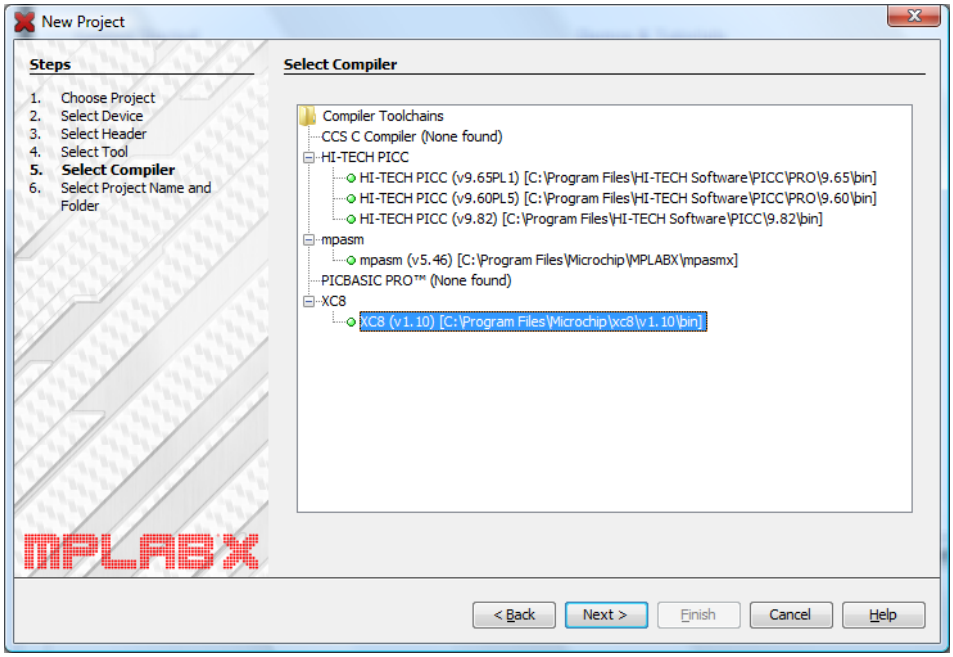


Рис. 18.14. Выбор транслятора с языка высокого уровня

Остаётся добавить имя проекта и его расположение. По умолчанию место, где будет расположен проект, создаётся при установке программы. Но вы можете указать папку для проекта, которая удобна для вас, как я указываю место, где будут собраны все проекты, относящиеся к этой главе.

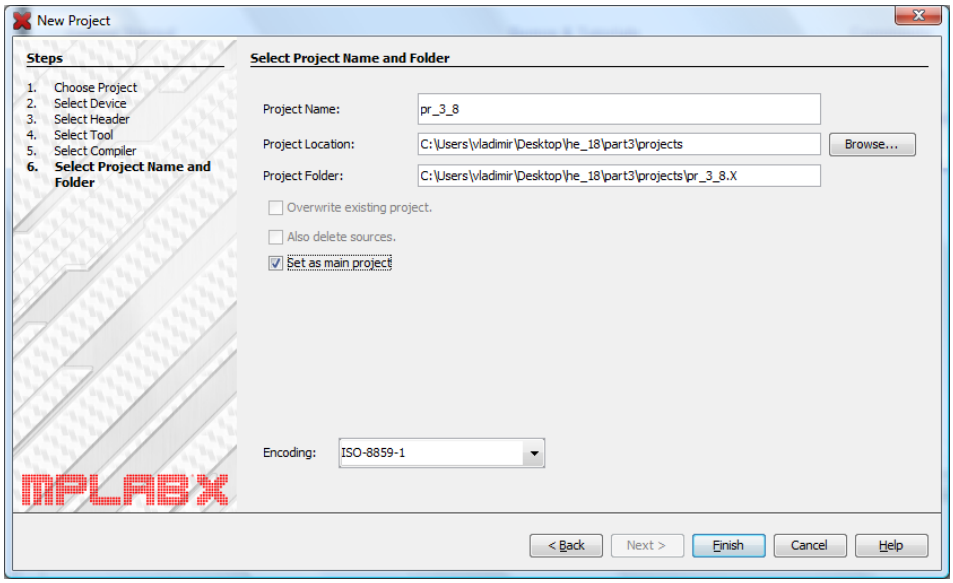


Рис. 18.15. Ввод имени проекта и выбор места его расположения

Остаётся нажать кнопку **Finish**, которой завершается создание проекта. Но завершается ещё не всё. Любая программа на языке Си должна иметь основной файл, где, собственно, и пишется программа. Чтобы сделать это в MPLABX, щёлкните правой клавишей мышки по разделу *Source Files* в окне навигации по проекту:

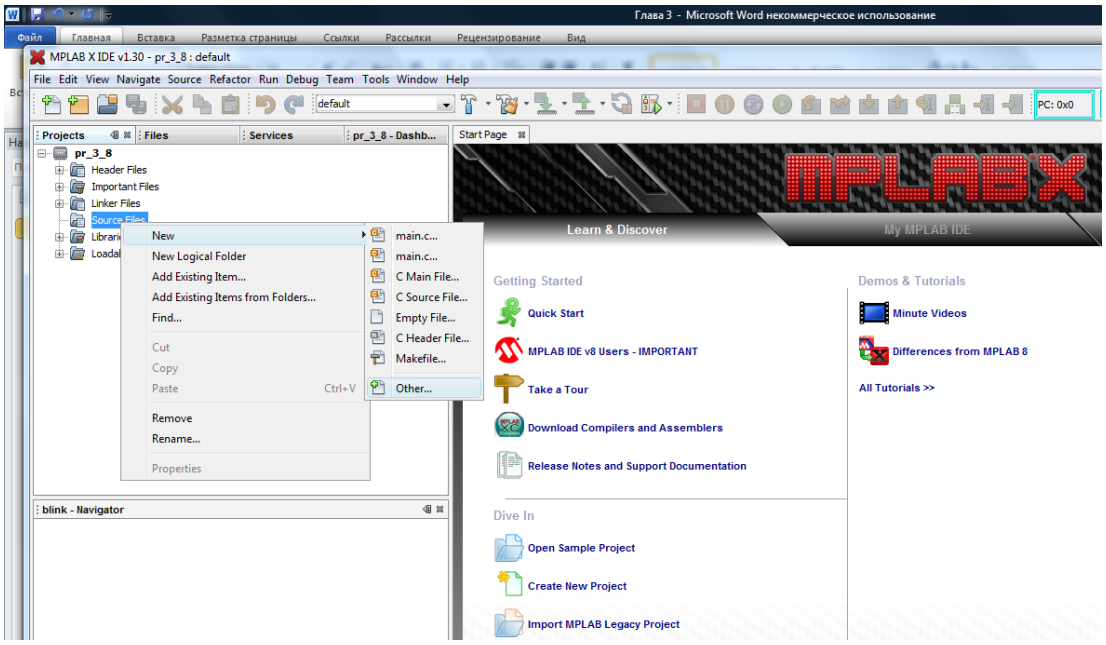


Рис. 18.16. Создание основного файла проекта

Выбрав New и Other, вы имеете возможность выбрать компилятор для этого файла и главный (или один из главных) файл проекта:

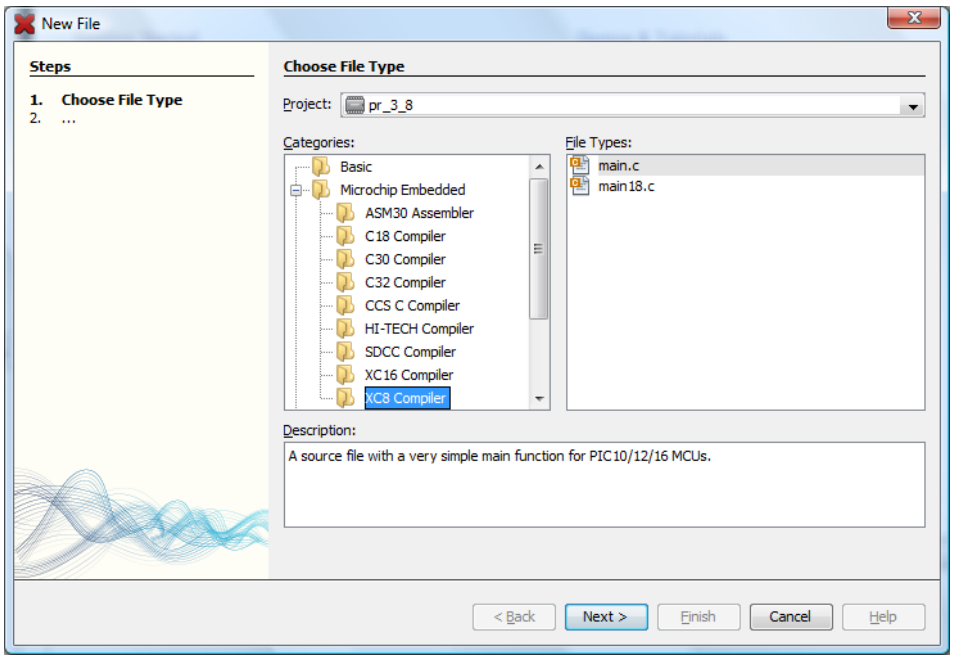


Рис. 18.17. Выбор главного файла проекта

После задания имени этого файла в следующем диалоговом окне, вы получите в редакторе текста заготовку:

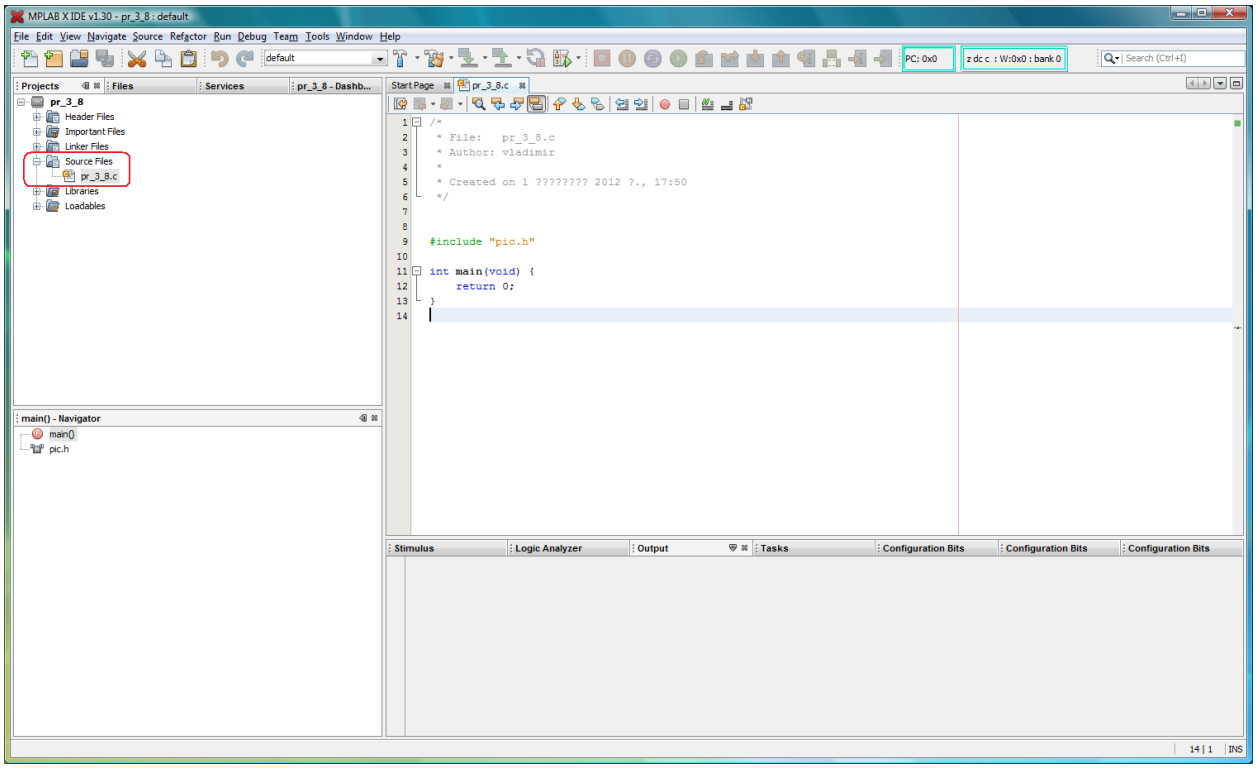


Рис. 18.18. Заготовка файла в окне редактора текста программы

В этом текстовом редакторе мы проведём достаточно много времени. Работать с текстом программы можно так же, как с обычным текстом – записывайте нужные команды, требуемые функции и т.д. Мы выяснили, что нам следует записать две команды нашей программы, которые и запишем:

```
/*
 * File:   pr_3_8.c
 * Author: vladimir
 *
 * Created on 1 ???????? 2012 ?., 17:50
 */


#include "pic.h"

int main(void) {
    TRISA = 0xFE;
    RA0 = 1;
    return 0;
}

```

Рис. 18.19. Первая программа

Обратите внимание на знаки вопроса в тексте шаблона. Это свидетельство непонимания программой кириллицы: в тексте программы, понимающей кириллицу, системная дата появилась бы как 1 сентября 2012 г. Обращайте на это внимание. Конечно, вы можете поэкспериментировать, но проблемы могут возникнуть не сразу. Не тогда, когда вы размещаете программу в папке, скажем, с именем «Проект» или тогда, когда вы называете проект «Мой проект». Хуже, если проблемы возникнут позже, вызывая ошибки при трансляции проекта. Некоторые программы не понимают кириллицу. Для них следует использовать любые имена: в пути к проекту, в названиях файлов, - написанные латиницей. И ещё одно, на что я хочу обратить ваше внимание – программа с именем main, главная программа проекта, в языке Си оформлена в виде функции, что особенно подчёркнуто в шаблоне для компилятора XC8: перед функцией main есть тип возвращаемых данных, это int, а в теле функции main есть запись return 0.

Проект можно было бы транслировать и проверить его работу. Но, если вы помните, я говорил о слове конфигурации, что важно для работы микроконтроллера. Хорошо сразу записать это слово конфигурации в исходном тексте программы. Для этого обратимся к пункту основного меню программы *Window*:

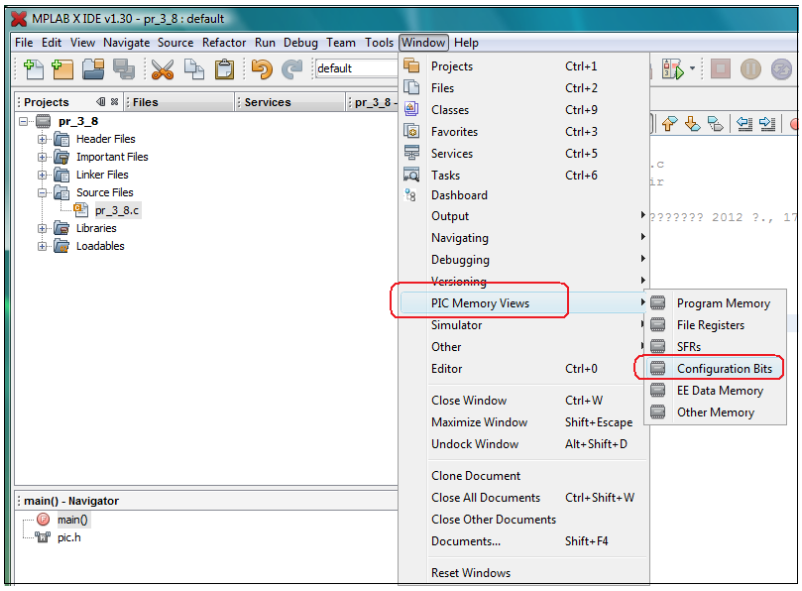


Рис. 18.20. Раздел задания слова конфигурации

Щёлкнув левой клавишей мышки по этому разделу, вы попадёте в окно задания слова конфигурации:

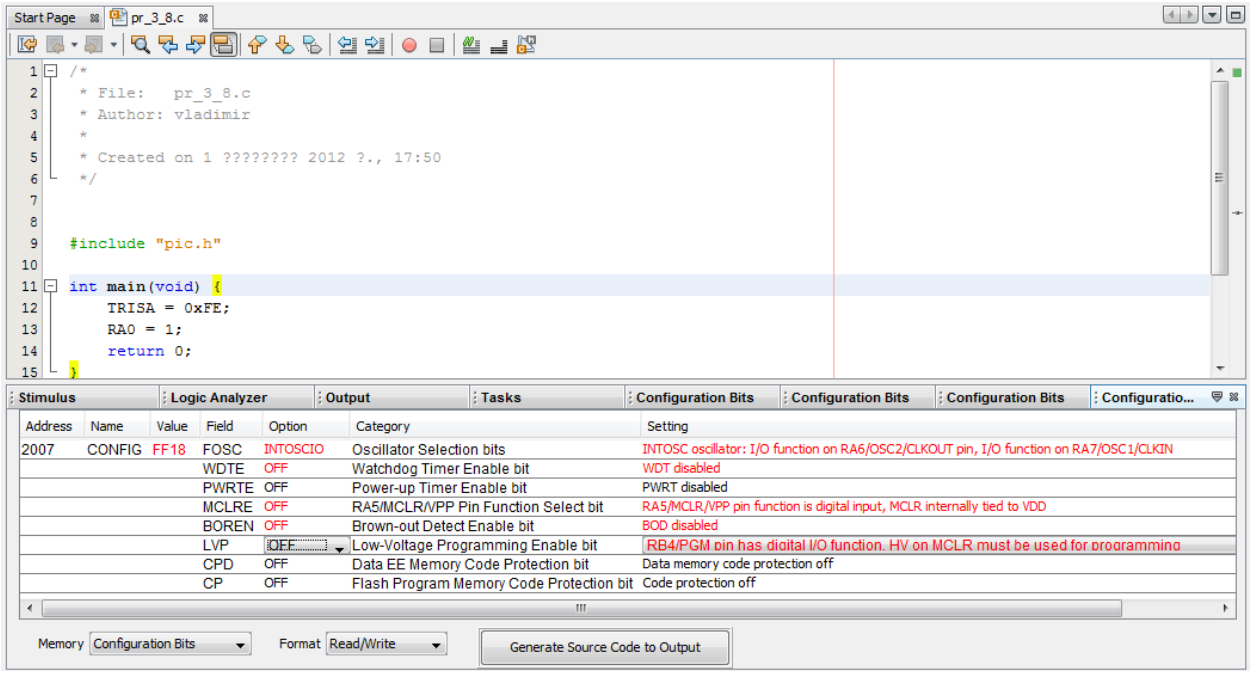


Рис. 18.21. Задание слова конфигурации

Каждый из битов слова конфигурации имеет стрелку, направленную вниз, справа от своего имени, в выпадающем списке после нажатия на эту стрелку, вы можете выбрать доступные решения. Я отключил все ненужные сейчас функции и выбрал внутренний генератор в качестве тактового генератора. Реальное слово конфигурации это 0x3F18 (шестнадцатеричное число), но первая тройка относится к максимальному числу для данной модели. Остаётся нажать кнопку Generate Source Code to Output, чтобы получить вывод слова конфигурации в этом окне:

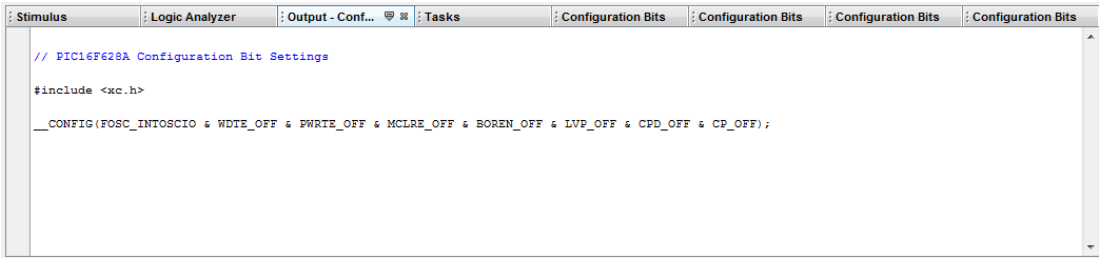


Рис. 18.22. Запись слова конфигурации для компилятора XC8

Остаётся скопировать это в буфер обмена и перенести в основной текст программы. Этот текст теперь выглядит так:

```
#include <xc.h>
__CONFIG(FOSC_INTOSCIO & WDTE_OFF & PWRTE_OFF & MCLRE_OFF & BOREN_OFF & LVP_OFF
& CPD_OFF & CP_OFF);

int main(void) {
    TRISA = 0xFE;
    RA0 = 1;
```

```
    return 0;
}
```

Эту программу можно оттранслировать и загрузить в микросхему или включить режим отладки. Чтобы транслировать программу, выбираем в основном меню программы раздел *Run* и *Build Main Project*.

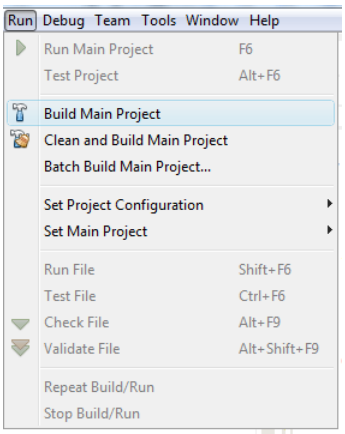


Рис. 18.23. Компиляция первой программы

Если в программе будут ошибки, то в окне вывода в нижней части экрана программы появится сообщение об ошибке. Если ошибок нет, то сообщение `BUILD SUCCESSFUL` завершает ваши усилия по созданию программы. Нет смысла отлаживать сейчас программу, она слишком проста. Но можно проверить, работает ли программа, используя доступные вам средства: другие программы, отладочную плату или макетную плату.

Мы написали первую, самую простую программу, но прошли все шаги от создания нового проекта до трансляции программы. В итоге, вы можете проверить, в папке с проектом появились отладочный и загрузочный файлы.

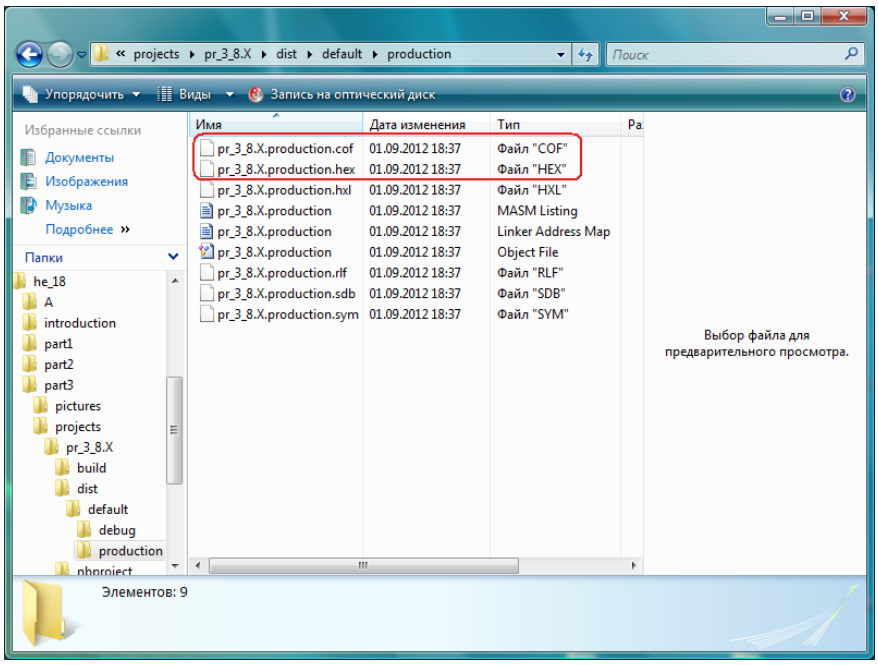


Рис. 18.24. Файлы для загрузки и отладки проекта

От воспоминаний к дню сегодняшнему (июнь 2021 г)

Загрузить среду программирования MPLABX можно с сайта производителя, достаточно в поисковике задать: *mplabx download*. Там же можно загрузить компилятор XC8. Конечно, скачивать следует версию для Linux. Установка осуществляется из терминала командами привычными в ОС Linux. Например, разместить распакованный файл с инсталлятором в домашнюю папку (директорию), задать в терминале полный путь от имени администратора, проверив, что распакованный файл является исполняемым:

```
/home/vladimir/xc8-installer.run
```

или

```
/home/vladimir/MPLABX-innstaller.sh
```

Всё остальное осуществляется аналогично установке в Windows.

После установки всего необходимого (программы и компилятора) можно создать новый файл, где и написать простую программу. Чем она проще, тем лучше для первой «пробы пера». Но есть один момент, который желательно усложнить. Поэтому я предлагаю такой вариант тестовой программы:

```
/*
 * File:   test.c
 * Author: vladimir
 *
 * Created on 27 ???? 2021 ?., 18:32
 */

// CONFIG
#pragma config FOSC = INTOSCIO  // Oscillator Selection bits (INTOSC
oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on
RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT
disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT
disabled)
#pragma config MCLRE = OFF      // RA5/MCLR/VPP Pin Function Select
bit (RA5/MCLR/VPP pin function is digital input, MCLR internally tied
to VDD)
#pragma config BOREN = OFF      // Brown-out Detect Enable bit (BOD
disabled)
#pragma config LVP = OFF       // Low-Voltage Programming Enable bit
(RB4/PGM pin has digital I/O function, HV on MCLR must be used for
programming)
#pragma config CPD = OFF       // Data EE Memory Code Protection bit
(Data memory code protection off)
#pragma config CP = OFF         // Flash Program Memory Code
Protection bit (Code protection off)

#define _XTAL_FREQ 4000000

#include <xc.h>
```

```
int main(void) {
    TRISA = 0xFE;
    while(1) {
    RA0 = 1;
    __delay_ms(1000);
    RA0 = 0;
    __delay_ms(1000);
    }
    return 0;
}
```

Саму программу я выделил цветом, чтобы показать, что она очень проста. Усложнения коснулись записи пауз. Дело в том, что разные компиляторы требуют разного написания. В данном случае команда начинается с двойного символа подчёркивания.

Кроме того, задание слова конфигурации осуществляется с помощью команды MPLABX в окне, как показано ранее, но само слово конфигурации записывается несколько иначе. И не забывайте, если вы используете паузу, сделать запись о тактовой частоте: `#define _XTAL_FREQ 4000000` (перед ключевым словом одно подчёркивание).

Когда текст программы написан правильно, её можно оттранслировать.

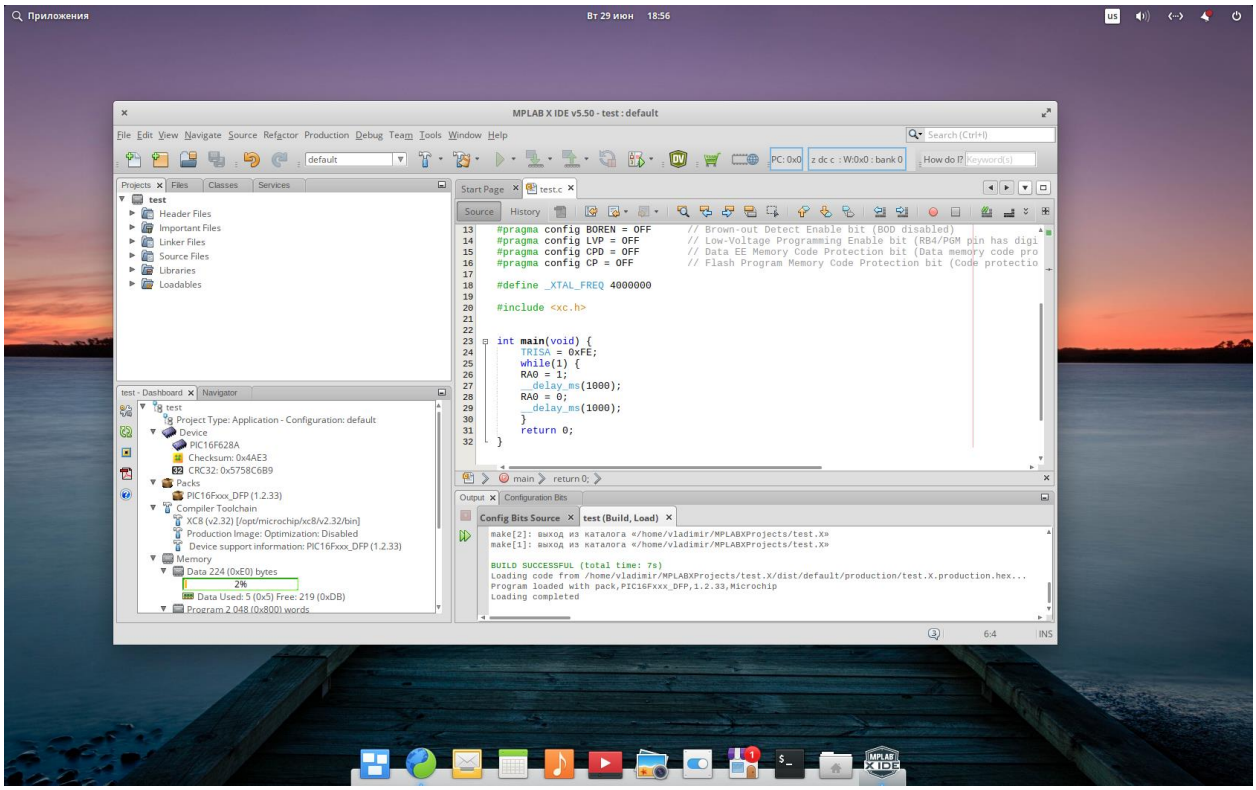


Рис. 18.25. Успешная компиляция программы

Об успешной компиляции сообщается, что она прошла успешно. Если есть ошибки, они будут указаны, будут даны рекомендации на что обратить внимание – достаточно прокрутить окно сообщений. Найти файл для загрузки, как я собираюсь это сделать, в схему SimilIDE, можно там, где вы храните все рабочие файлы MPLABX, где есть папка dist, в которой можно найти (при желании) hex-файл (и не только).

В программе SimulIDE достаточно добавить на рабочее поле микроконтроллер PIC16F628A. Но рабочий hex-файл лучше отправить в домашнюю папку, поскольку приходится вручную прописывать его в свойствах микроконтроллера, а, чем короче путь, тем быстрее будет результат.

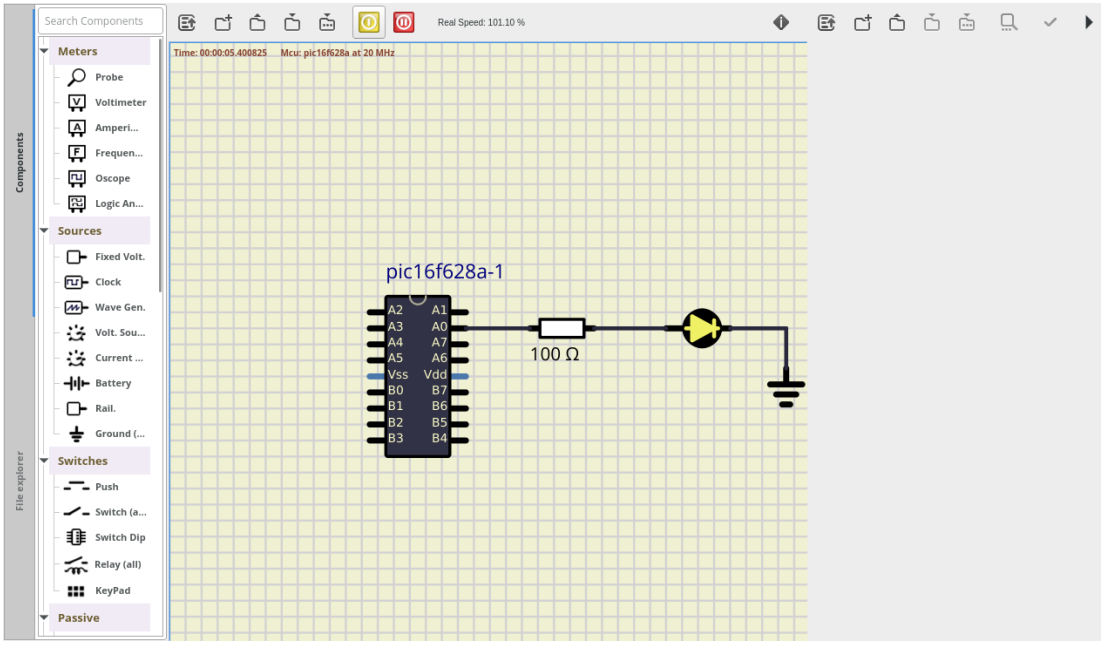


Рис. 18.26. Моделирование программы в SimulIDE

Особенно полезно моделирование в среде SimulIDE тогда, когда вы проверяете работу с компонентами, которые не предусмотрены, например, программой MPLABX. Скажем, вы добавили транзистор для управления светодиодом, требующим большого тока при включении.

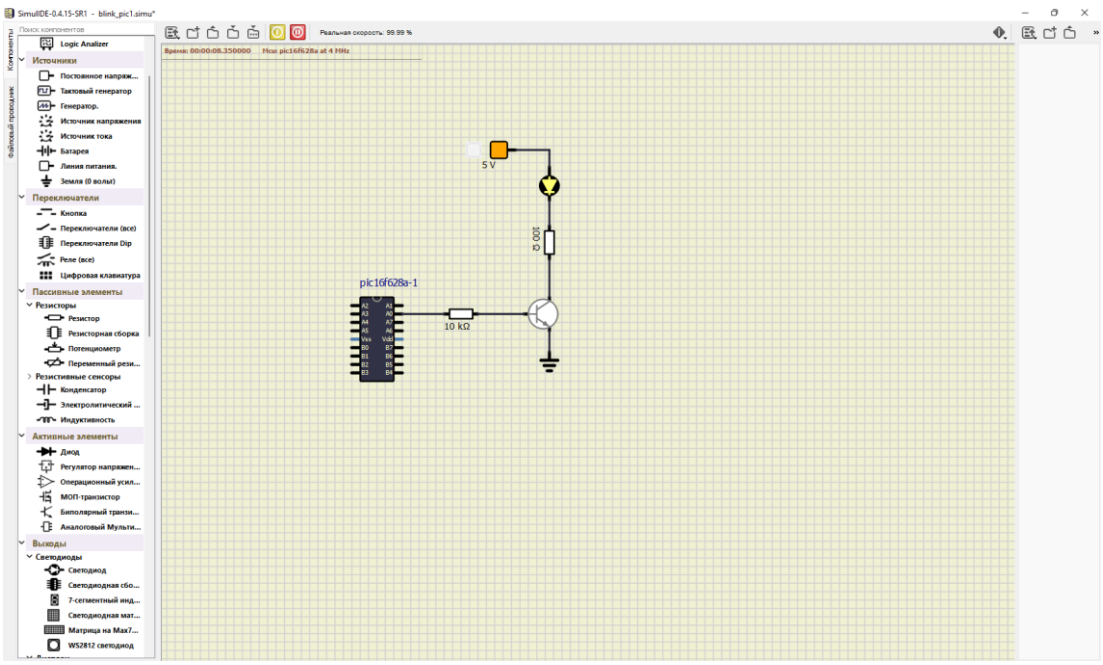


Рис. 18.27. Использование транзистора для усиления выходного тока контроллера

Резюмируя сказанное

Операционная система Linux имеет множество дистрибутивов, которые могут работать и на самых последних моделях компьютеров, и на достаточно старенька их собратьях. Если ваш компьютер, как, например, мой не подходят для Windows 11 (или не поддерживают даже последние сборки Windows 10), вы всегда можете использовать подходящий дистрибутив Linux. По возможностям и удобству использования он не будет уступать Windows.

Рис. 18.28. ElementaryOS в Windows11

Я не уверен, что проверил работу всех бесплатных программ, которые могут быть интересны радиолюбителям, это так, но с интересами дело обстоит обособлено. Сочиняя разные истории, я старался придумать примеры, способные заинтересовать начинающих. Однако уверенности, что это так, у меня никогда не было – что кажется интересным тебе, будет неинтересно другим. Поэтому, я очень на это надеюсь, начинающие любители будут сами ставить посильные задачи, которые им интересно решить, используя программы, о которых рассказано здесь, о которых я рассказывал в других историях, и которые любители отыскали самостоятельно без моих подсказок.

P.S. Не удержался…

Заработавшая под wine программа QucsStudio, заставила меня задуматься, а не будут ли работать другие программы. Этому способствовала статься об установке MS Office 2010 в Linux, на которую я наткнулся в Яндекс.Дзен. Я, как и советовали в статье, скачал и установил пакет PlayOnLinux, и попробовал установить Flowcode 4. Но быстро выяснил, что установить можно то, что включено в список PlayOnLinux (или что-то я не понял).

Тогда я в терминале запустил установку Flowcode 4. И установка прошла. Правда не получился запуск с помощью ярлыка в списке приложений (или я не дождался запуска), тогда я использовал проводник wine, который можно запустить из терминала командой: `wine explorer`.

На диске C: в разделе программ для 32-битовых приложений нужные рабочие файлы запускаются двойным щелчком, как это происходит и в Windows.

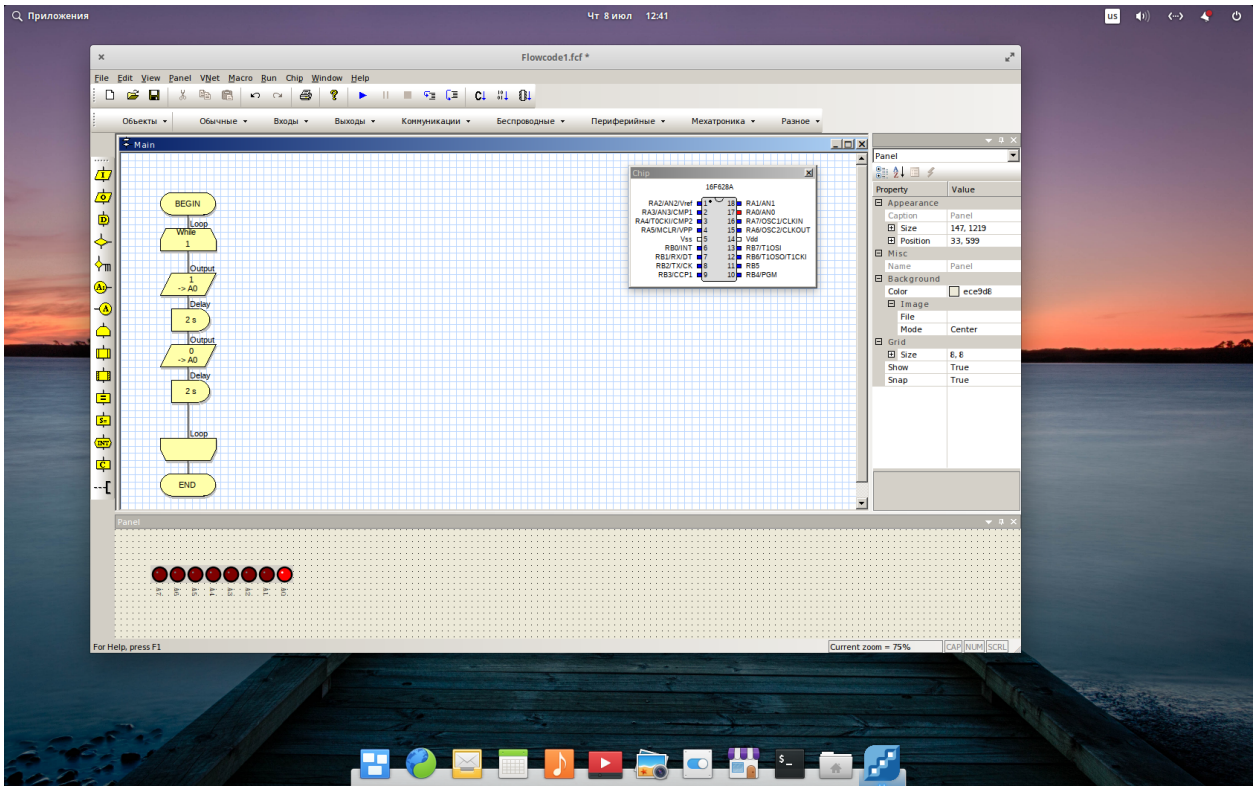


Рис. 18.28. Работа с программой Flowcode 4 в ElementaryOS

Программа собралась, моделирование прошло, и компиляция дала все требуемые файлы. Не помню, были отдельные небольшие нестыковки в Flowcode 4 или нет, но работать можно. Правда, не мешало бы проверить работу с программатором, не мешало бы проверить работу с более сложной программой, однако работает, и можно надеяться, что будет работать.

Полученный hex-файл мне захотелось проверить, подключив резистор и диод (как положено). Я вспомнил о программе Proteus. Установка тоже прошла, но я сразу столкнулся с проблемой:

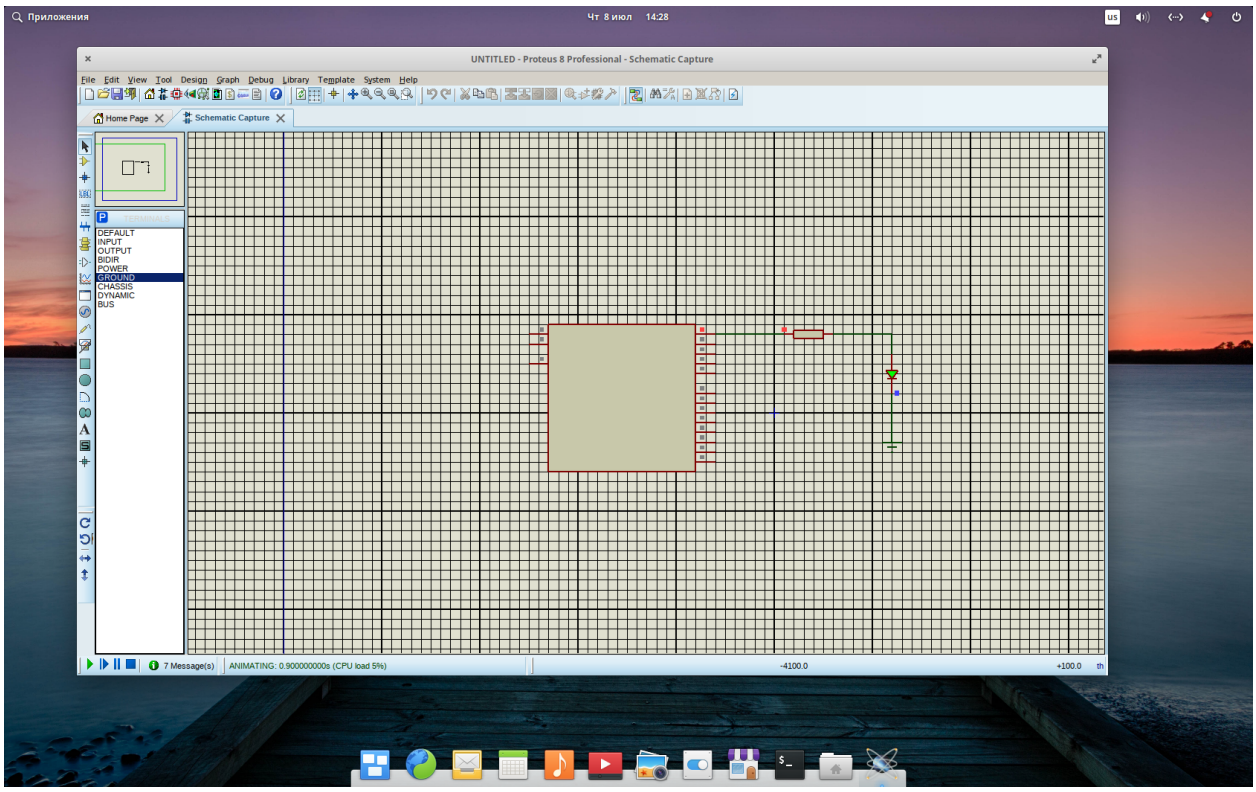


Рис. 18.29. Появление проблем со шрифтами в Proteus

Надписи не появились только в рабочем поле. Видимо, проблему следует решать глобально, но я использовал локальное решение: Template->Set Text Style, где выбрал для нескольких позиций: PIN NAME, COMPONENT ID, COMPONRNT VALUE, - шрифт Tahoma.

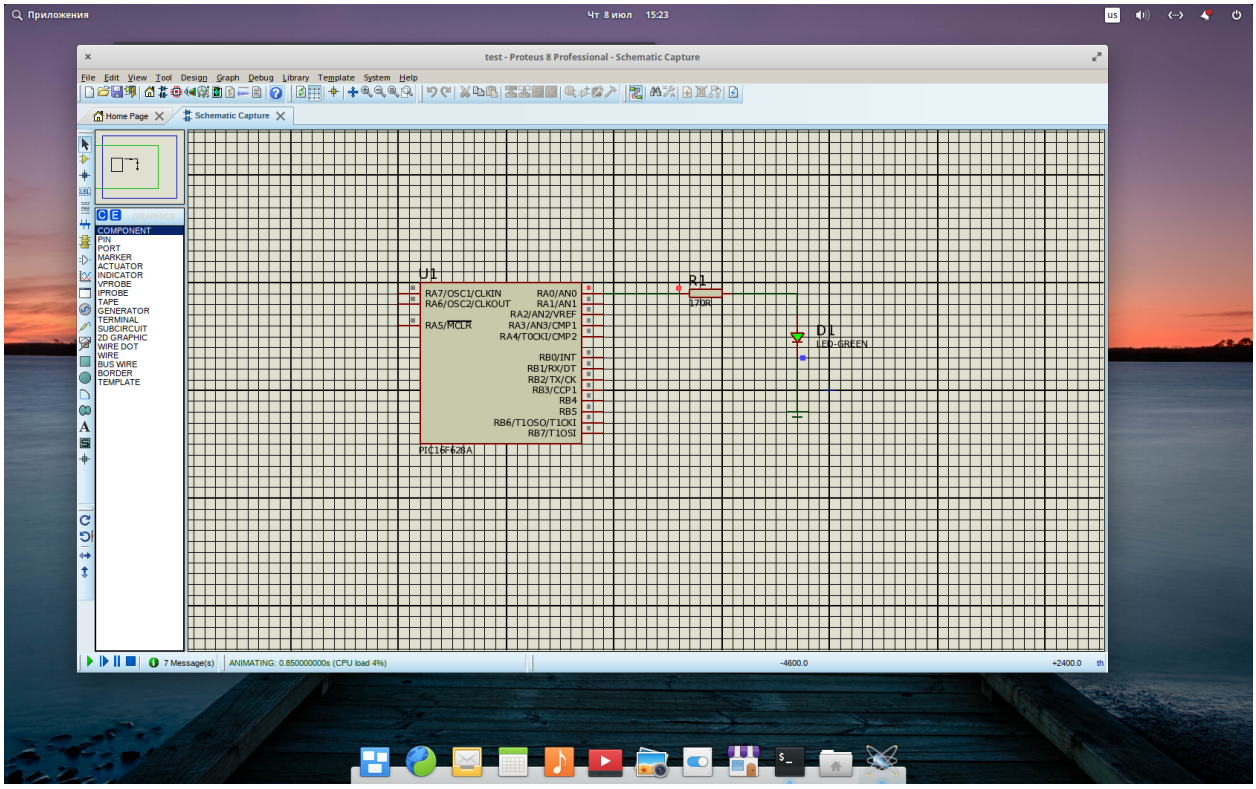


Рис. 18.30. Проверка hex-файла из программы Flowcode

Если сохранить файл, то при следующем запуске всё сохранится, но для нового файла придётся заново указывать шрифт. Возможно, это не единственная проблема, но беглая «проба пера» с осциллографом показала, что он работает. Есть с чего начать, если у вас есть возможность и желание использовать эти программы. Я не исключаю, что можно поставить и другие Windows-программы.

P.P.S. Вот, забыл, получится или нет, но надо попробовать…

Толи жара тому виною, толи экспериментальная 11 версия Windows, толи, наконец, моё нетерпение, когда, не дожидаясь завершения внутренних разборок в чёрном ящике VirtualBox, я перезапускаю ElementaryOS, но с каждым днём такая работа раздражает всё больше. Приходится подолгу ждать завершения загрузки, приходится ждать, когда можно что-то сделать. Конечно, нужно хорошо настраивать VirtualBox, не следует перегружать виртуальную операционную систему, всё правильно. Но я вспомнил, что не лишне было бы проверить работу программы Flowcode со встроенными в микроконтроллер модулями. Это не так долго, так мне казалось...

Первые грабли, о которых я хочу сказать – это установка и удаление программ в wine. Установку можно осуществлять из терминала, можно из проводника, если срабатывает, но лучше всего использовать проводник Windows (из wine), который запускается из терминала командой `wine explorer`. Но об этом чуть позже.

Итак, небольшая программа для отправки символа «1» (0x31 шестнадцатеричное) по встроенному последовательному порту (UART).

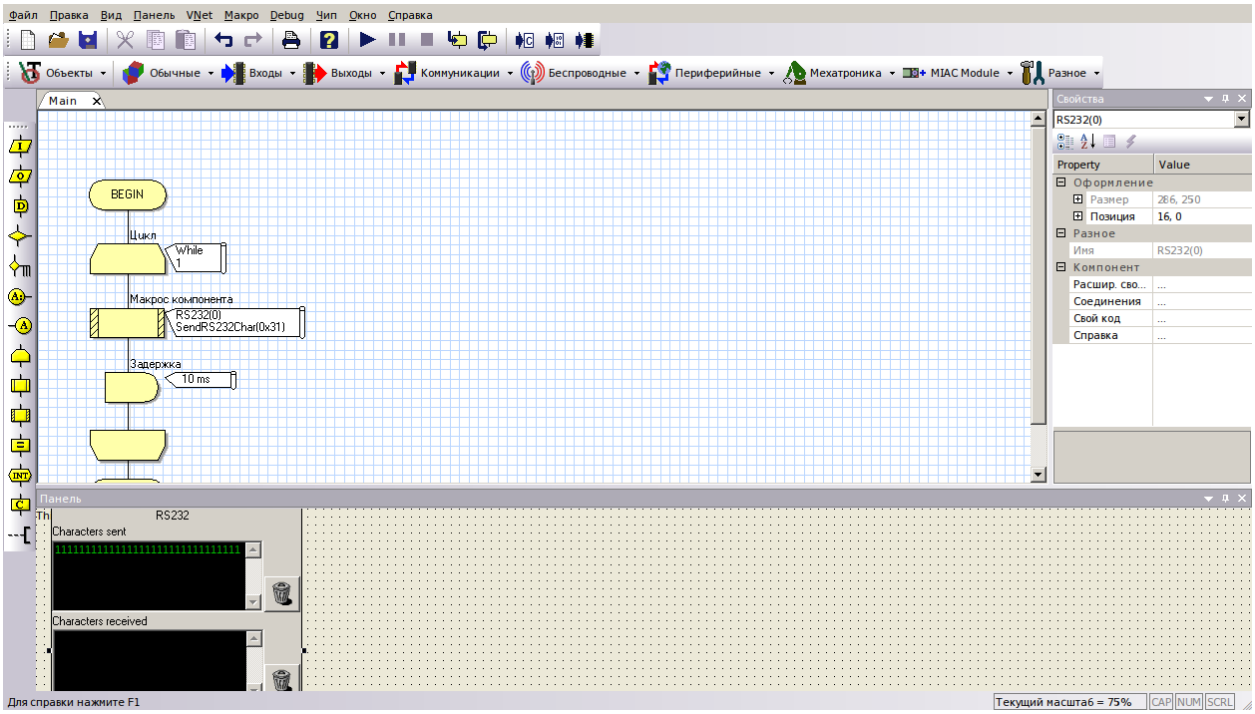


Рис. 18.31. Программа отправки «1» по UART (RS232)

В настройках не получилось задать тактовую частоту (отсылаю к абзацу про толи жару…), но в остальном программа работает, моделируется, программа компилируется. Полученный hex-файл вполне работает с программой ISIS (Proteus).

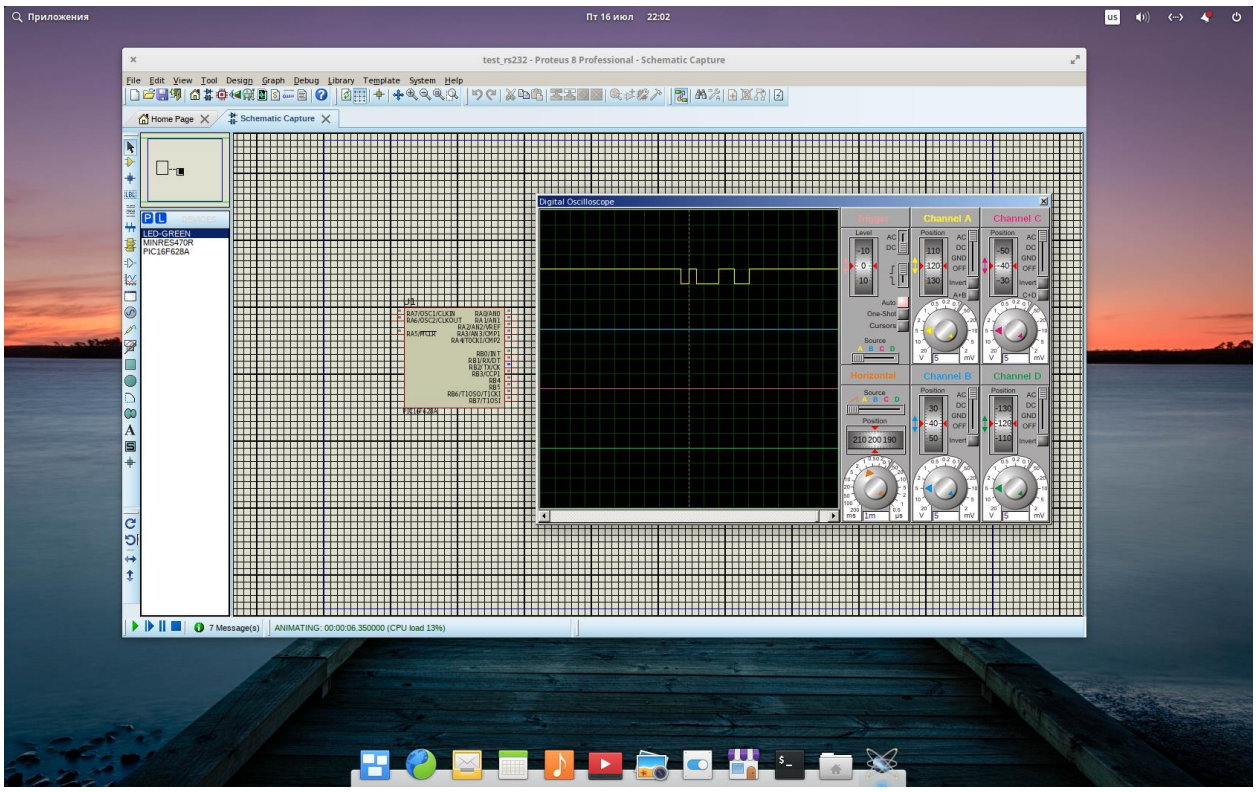


Рис. 18.32. Работа микроконтроллера по отправке символа через UART

По виду отправляется правильное значение, то есть: 00110001. Таким образом, программа Flowcode работает в среде wine, что мне и было интересно. А пока я возился с этим «не так долго», я вспомнил ещё об одной полезной начинающим программе. Её можно скачать, как это сделал я, но она в форме установщика. И так пришло время вернуться к граблям, вернее к обходному пути этого знакового устройства, граблей.

Запустив проводник, выбираем в левом окошке раздел «Мой компьютер».

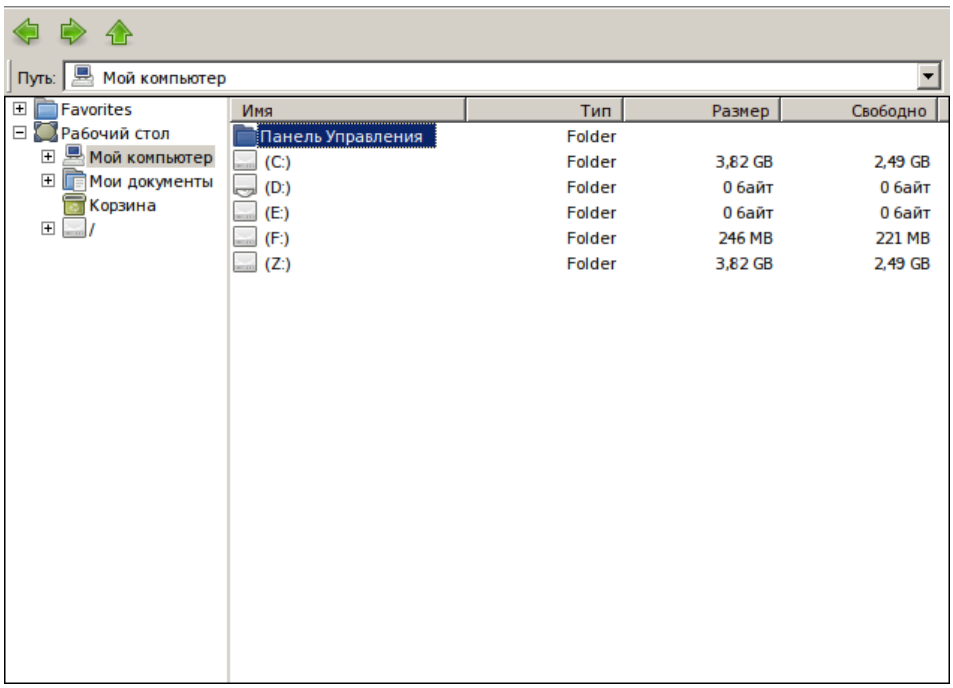


Рис. 18.33. Проводник в среде wine

В правом окошке есть «Панель упрвления», которую следует открыть.

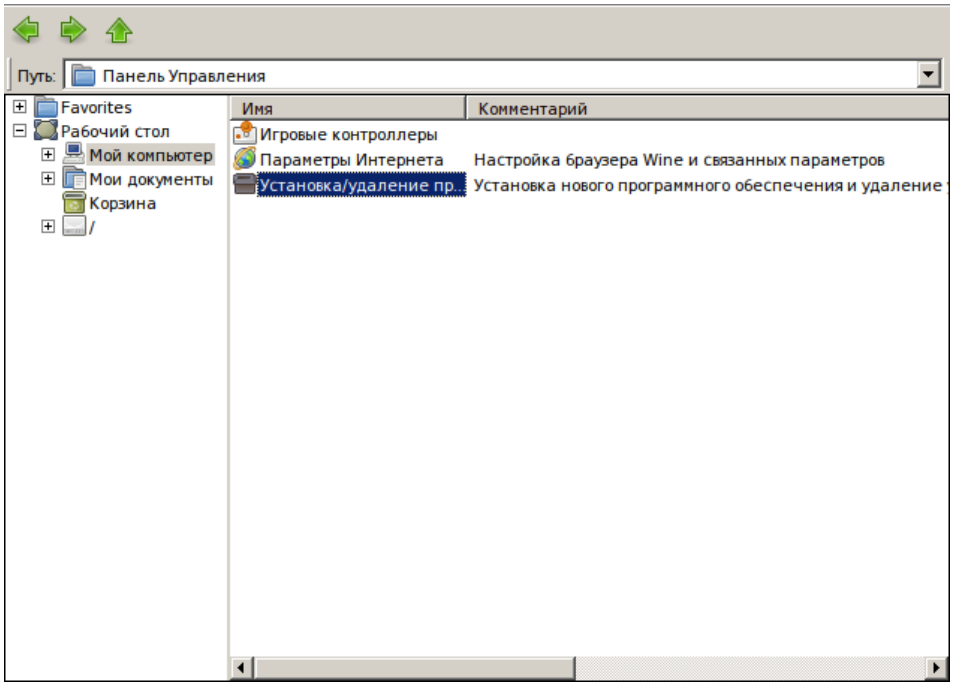


Рис. 18.34. Панель управления в проводнике wine

Для установки и удаления программ есть раздел «Установка/удаление программ», которым и следует пользоваться для обхода граблей. Если его открыть, то можно увидеть кнопку, которой следует пользоваться при установке, **Установить**. Установить можно файлы `exe`, но, как в нашем случае, установщиком файлы с расширением `msi`. Сам файл может иметь, наверное, разные имена, у меня получился файл с названием electronica. Но программа устанавливается, программа работает.

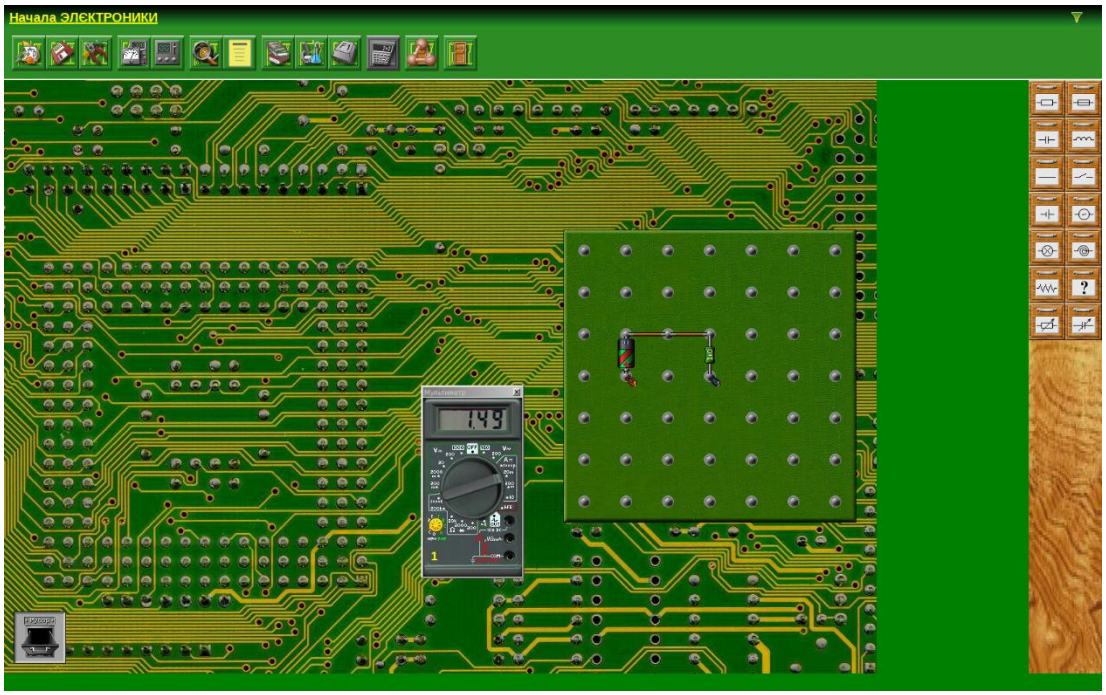


Рис. 18.35. Работа программы «Начала электроники» в ElementaryOS (Linux)

В виртуальном пространстве с возможностями моего компьютера работать не так комфортно, как это было бы в реальном, но это и не работа, это просто короткий рассказ, который окончательно подошёл к концу.