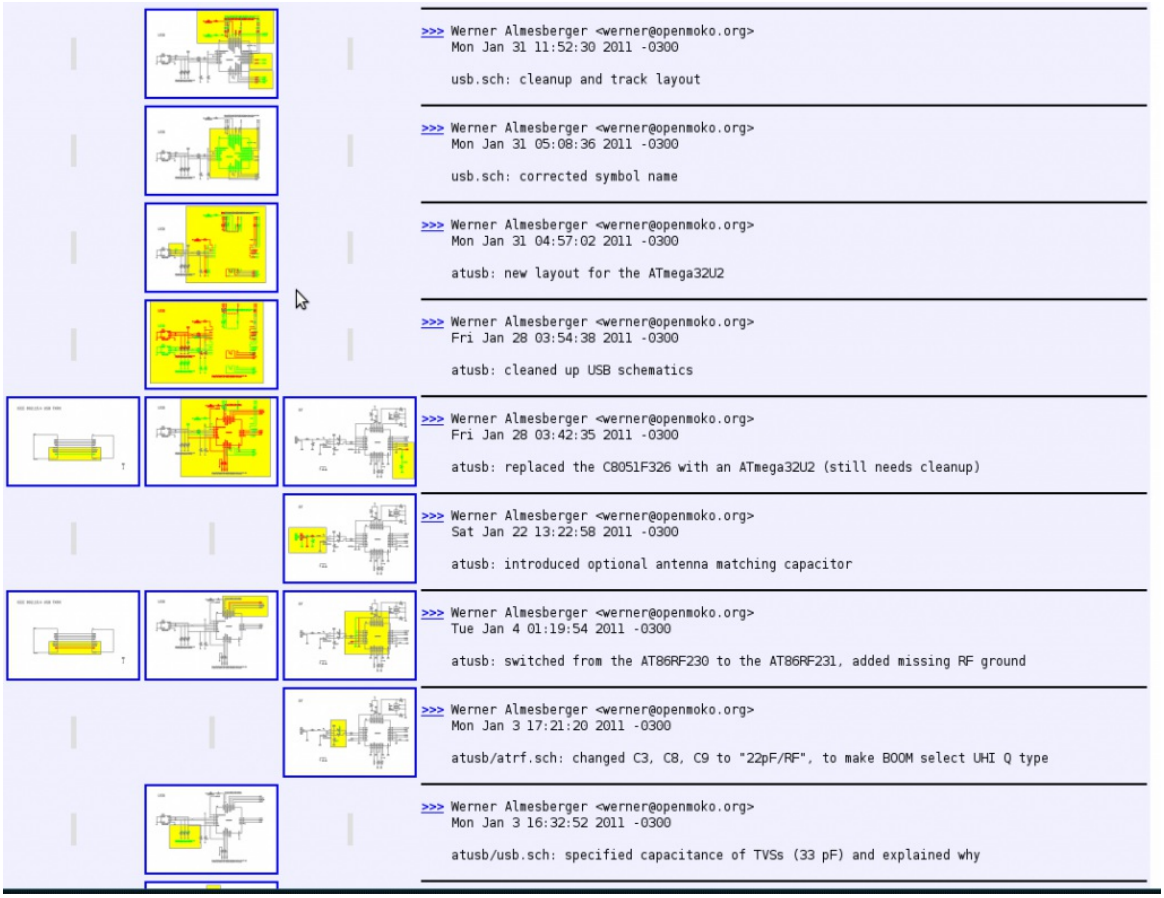
Как известно KiCAD хранит данные в текстовом формате, а чем хорош текстовый формат? Текстовый формат можно редактировать вручную любым текстовым редактором, а также текст очень удобен при контроле версий. Мы уже говорили о git , а также о том как можно хранить базу своих библиотек.



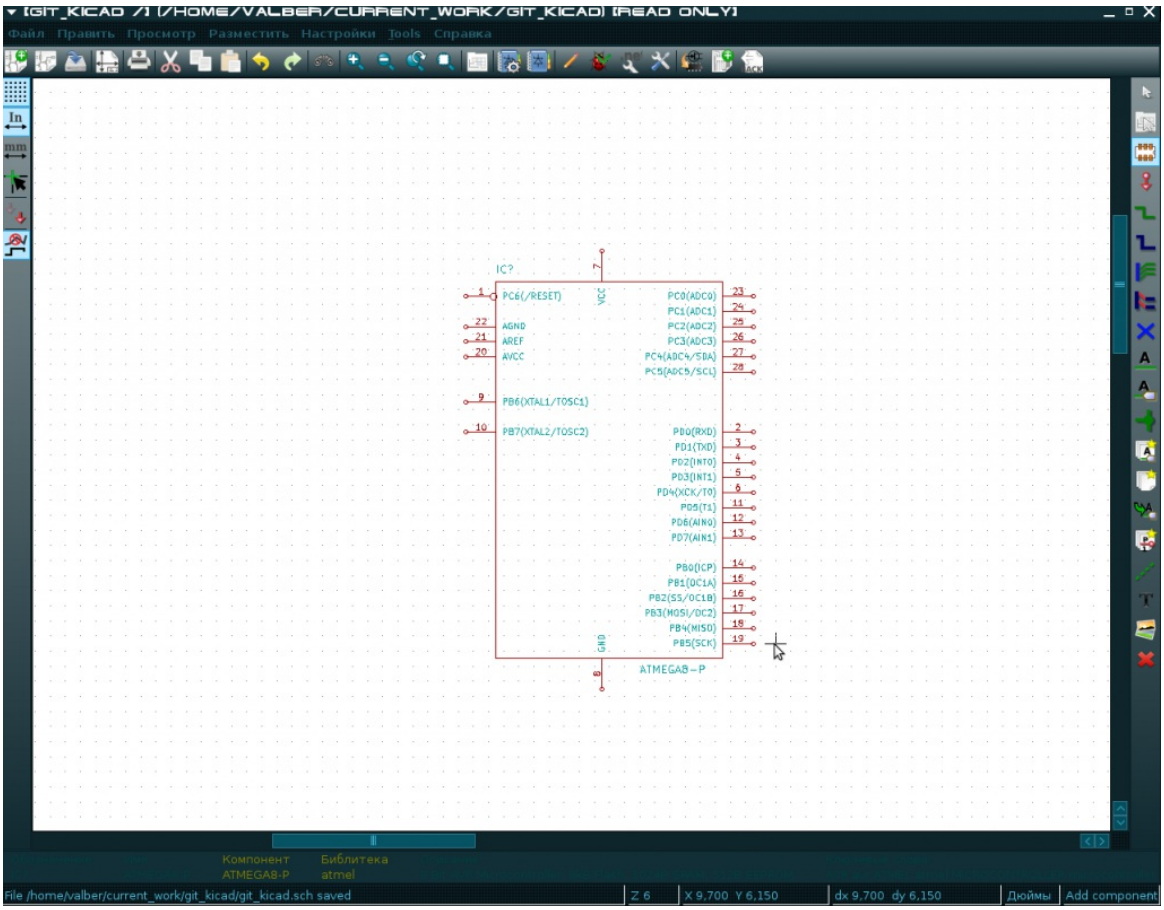
А теперь мы поговорим об инструменте разработанным Werner Almesberger из Qi Hardware позволяющую создавать графическую историю изменений вашей схемотехники, основанную на истории изменений в git(пример работы). Если вы вели разработку какого-нибудь устройства, и следовали старому правилу, что надо часто сохраняться , сохраняться под разными именами , если делаешь модификацию, то нужно создать ответвление , отдельную папку. В конце получается море каталогов в которых вы ориентируетесь только пока вы работаете над проектом.

Использование Git для ведения проектов в KiCAD

А что если мы пойдем другим путем и в начале разработки инициализируем в папке проекта git?

?

```
1 mkdir our_prj
2 cd our_prj
3 git init
```

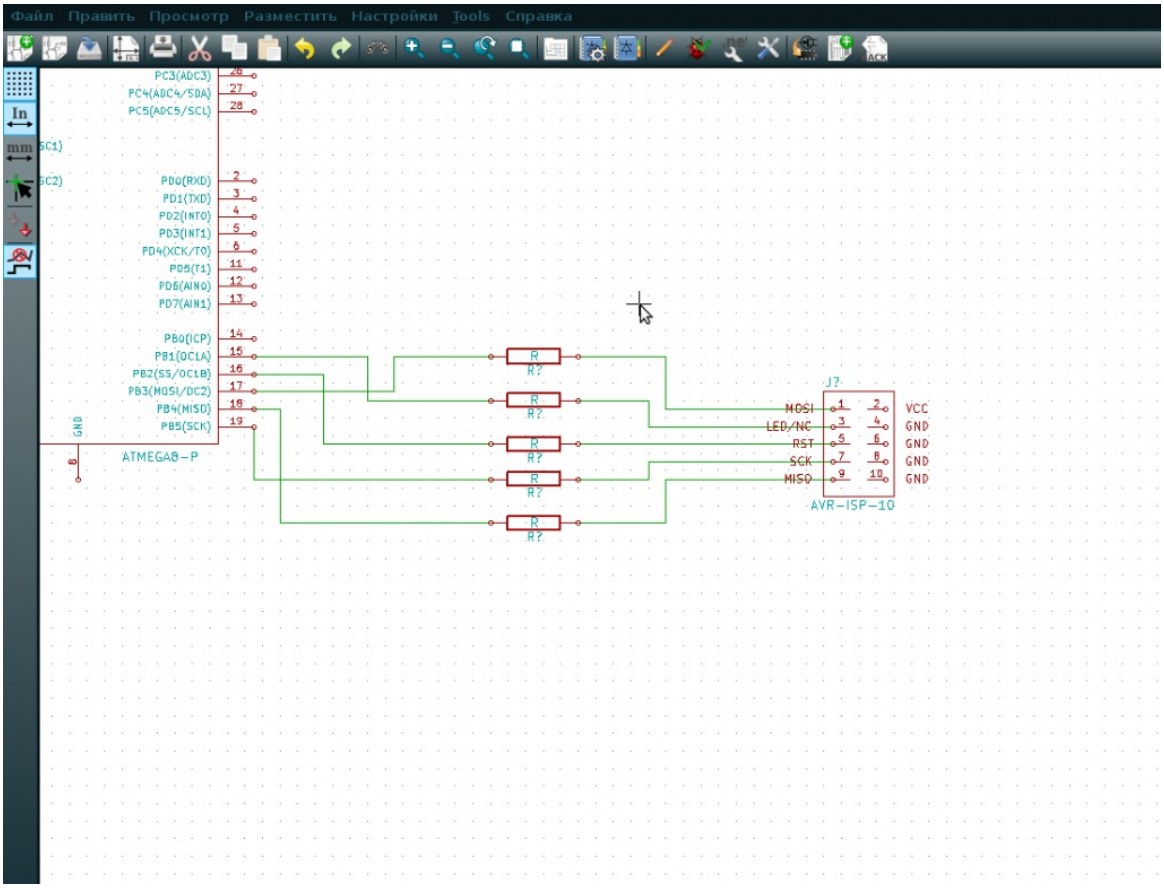


Пример создания коммита, с начало добавляем файлы, изменения в которых надо учитывать, а потом фиксируем их. Например вот так:

?

```
1    git add our_prj*
2    git commit -am "first_step_for_human"
```

а вот теперь мы произведем изменения и создадим второй коммит



посмотрим список последних коммитов

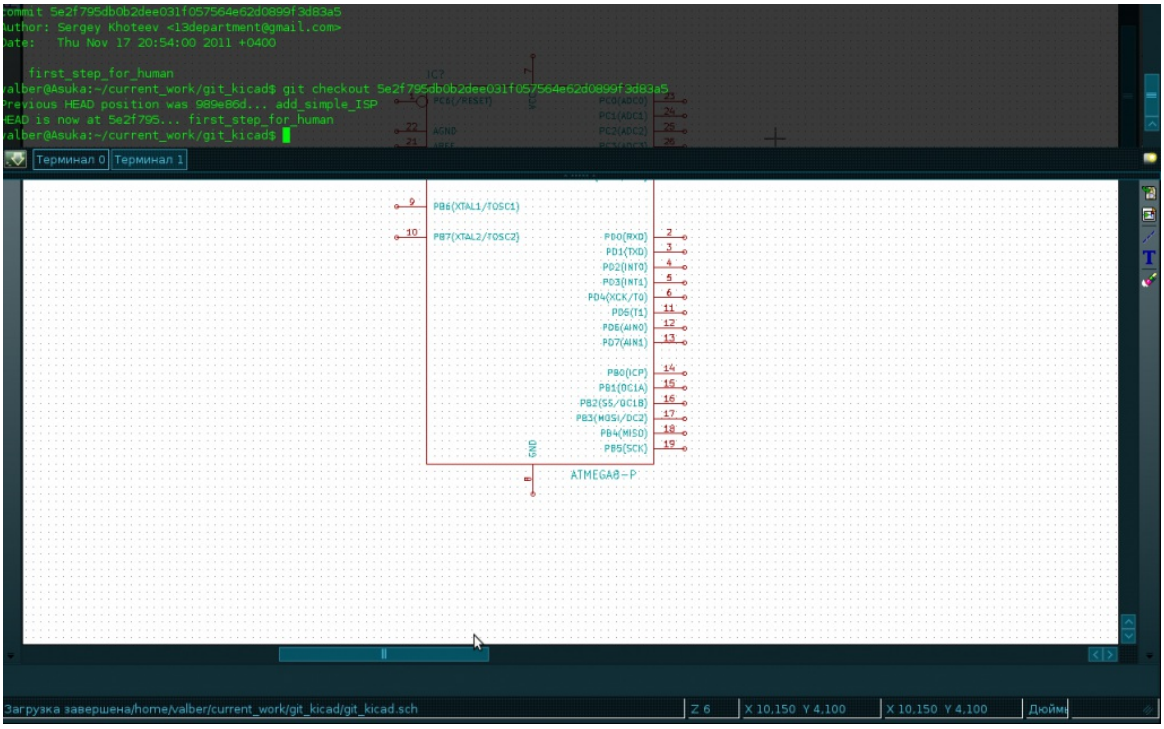
?

```
1    git log
```

прекрасно, но допустим мы хотим откатить изменения обратно
Пример отката версии, по 14-значному номеру коммита

?

```
1    git checkout  **номер коммита***
```

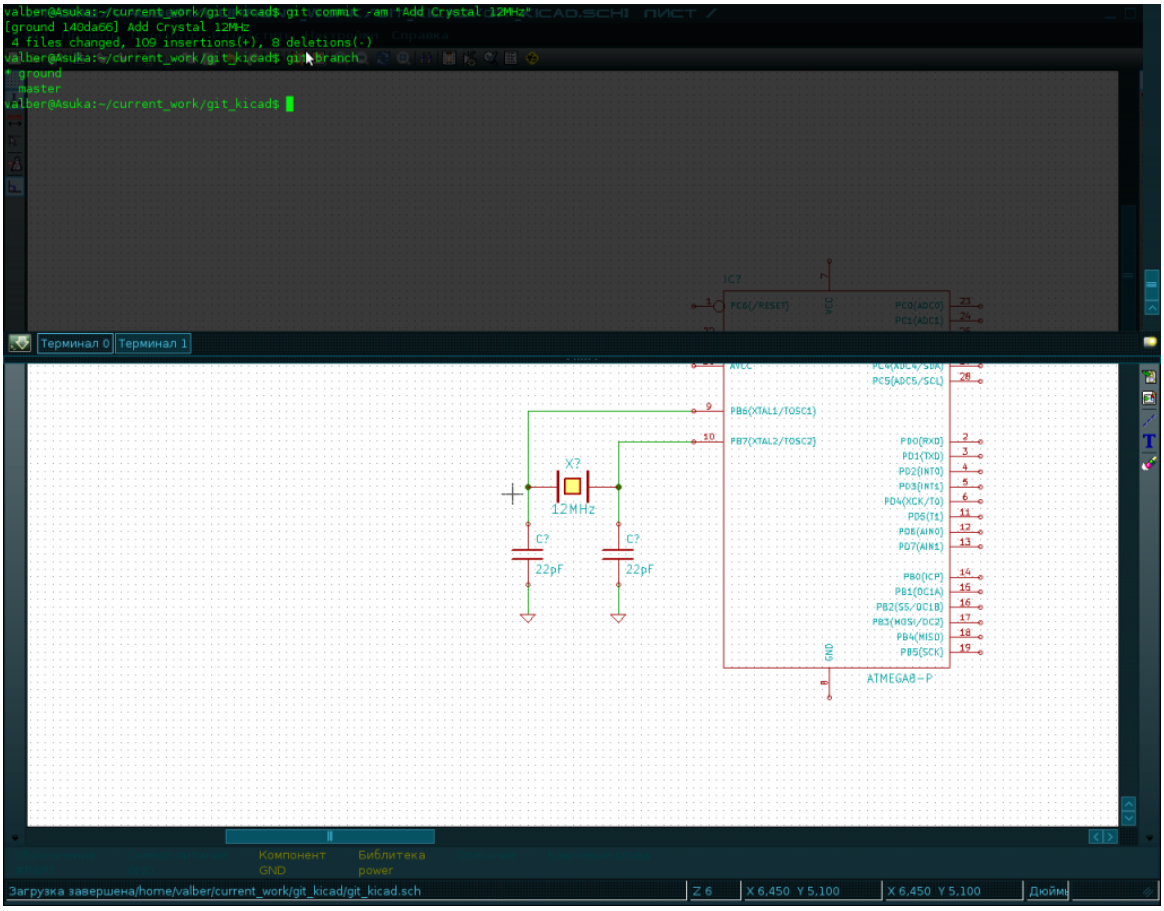


И допусти в этот момент мы решили что пора начать вести совместную разработку или просто отдельный блок схемы, обрабатывать в отдельной ветке. Так давайте её создадим(имя ей ground)!!

?

1 git branch ground
2 git checkout ground

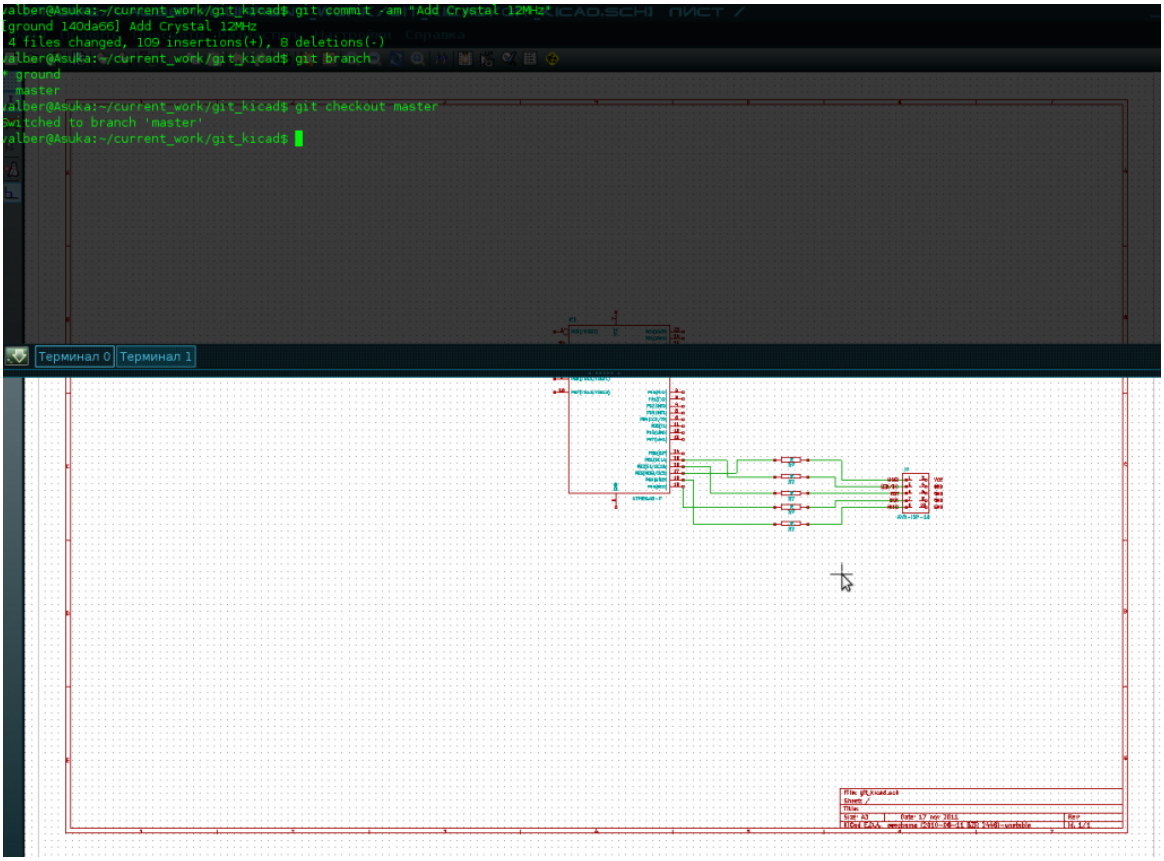
Теперь давайте что, нибудь поменяем и создадим коммит в этой ветке!



Теперь вернемся к основной ветке и посмотрим

?

1 git checkout master



Теперь самое интересное, это слияние веток, и хотя мы сделали допущение т.е. работа у нас велась раздельно и области её не перекрывались, а также мы не двигали наш микроконтроллер, который является основой обеих веток. Итак сливаем изменения из ветки ground в ветку в которой сейчас находимся!

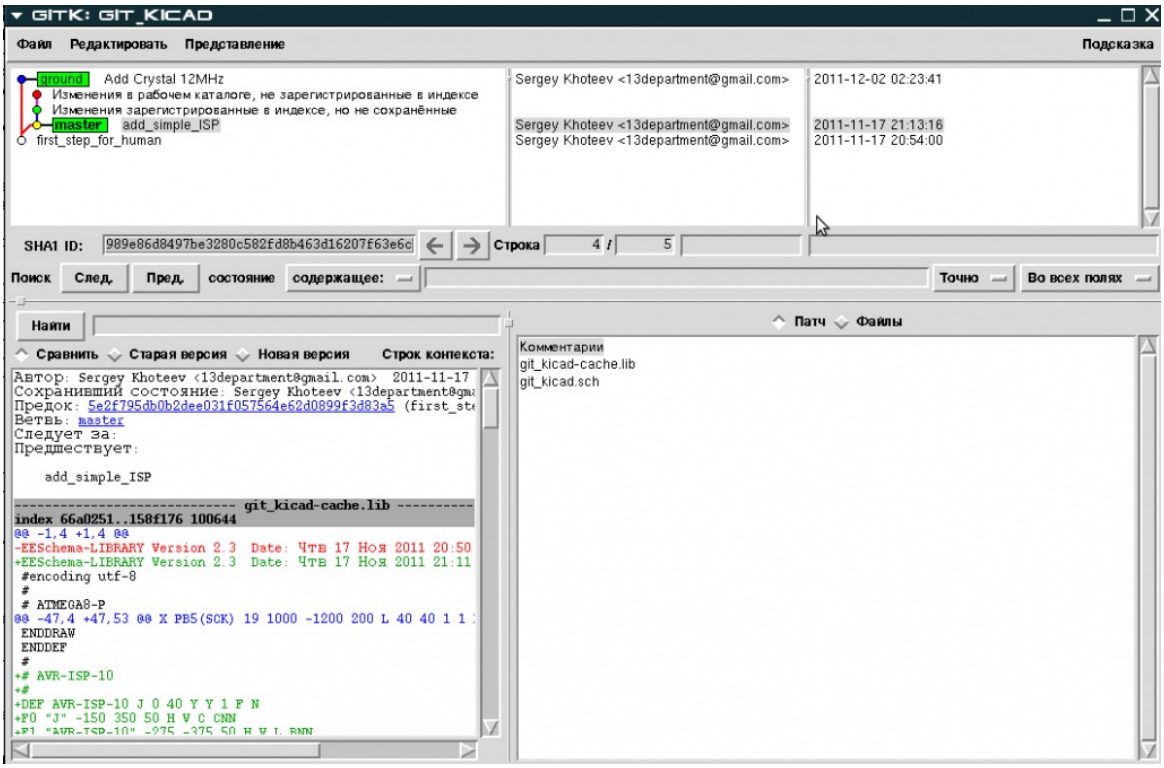
?

```
1  git merge ground
2
3  Auto-merging git_kicad-cache.lib
4  CONFLICT (content): Merge conflict in git_kicad-cache.lib
5  Auto-merging git_kicad.sch
6  CONFLICT (content): Merge conflict in git_kicad.sch
7  Automatic merge failed; fix conflicts and then commit the result.
```

Увы, автоматически не получилось соединить наши ветки. Для повышения умения, читаем руководство и разбираемся как разрешать конфликты.
В графическом интерфейсе и с цветами будет наверно удобней читать отчет о конфликтах.

?

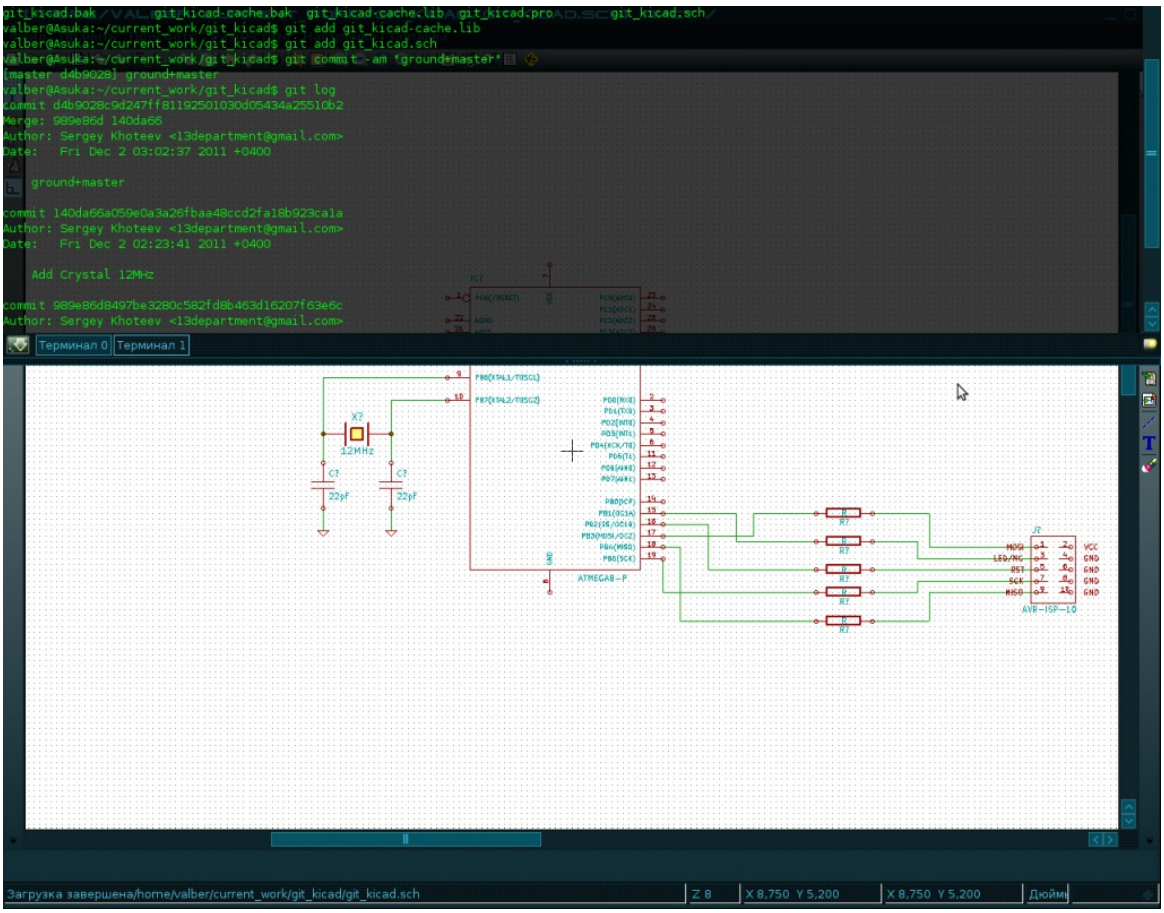
```
1  gitk --merge
```



Ещё справка/пример. Здесь я не буду комментировать, это деликатная задача, один раз сделаете поймете(но если все же будут вопросы, пишите), просто посмотрите как устроен формат .sch на википедии или в нашей статье. Итак у нас была два файла с конфликтами , исправив их , мы учитываем изменения в них и создаем коммит!

?

```
1	git add git_kicad-cache.lib
2	git add git_kicad.sch
3	git commit -am "ground+master"
```



Хотя с помощью определенных программ вы можете построить граф разработки. Но это все операции с текстом, мы же хотим чтобы это было наглядно и вот для этого и существует программа schhist. Возможно вы уже заметили что на скриншотах, у меня изображена старая версия kicad, все дело в том что для работы schhist необходимо чтобы eeschema могла работать и генерировать картинки различных форматов из командной строки,для этого были введены изменения в код ревизии 2448, но к сожалению в основное дерево эти изменения не добавлены. Так что вам придется выкачать старую версию kicad , пропатчить её и установить.

Подготовка kicad к работе с schhist

Примечание: я добавлял только патчи, от разработчика Werner Almesberger, остальные изменения не получилось добавить, подробнее тут.

Недавно[29.1.2012] разработчики обновили патч до версии kicad 3378 ,теперь сборка стала еще проще, сначало качаем необходимые патчи и программы от Qi-Hardware

?

```
1	git clone git://projects.qi-hardware.com/eda-tools.git
```

Затем выкачиваем версию kicad 3378 и собираем её, при пересборке пользуйтесь инструкциями

?

```
1	bzr checkout -r 3378 lp:kicad kicad.bzr
2	cd kicad.bzr
```

```
3  ln -s путь_к/eda-tools/kicad-patches patches
4  quilt push -a
5  cmake -DKICAD_TESTING_VERSION=ON .
6  make -j 5
7  sudo make install
```

Установка schhist

Здесь описано более подробно, т.к. у Qi-Hardware это программа работает на сервере и создает карту работ всех разработчиков. Всё что надо вы уже скачали с помощью git , как eda-tools

?

```
1  cd eda-tools/schhist/ppmdiff && make
```

Применение schhist

В папке /eda-tools/schhist вы найдете

- **schhist2web** — это наш основной инструмент, он задействует сам работает с другими утилитами, на выходе создает html страницу с историей схем
- **schps2pdf** — создает pdf файлы из с генерированного eeshema ps файла
- **schps2ppm** — создает ppm файлы из с генерированного eeshema ps файла
- **subschname2file** — Переводит название подсхем(видимо из истории git), в название схем
- **sanitize-schem** — Удаление из схем элементов которые могут вызвать нарушение работы
- **sanitize-profile** — Удаление из проекта элементов которые могут вызвать нарушение работы
- **gitsch2ps** — создает ps файлы для kicad из истории изменений git
- **normalizeschps** — нормализует ps файлы созданные eeshema
- **gitenealogy** — отслеживает происхождение фалов даже если они были переименованы
- **gitwhoareyounow** — также используется для нахождения переименованных фалов
- **ppmdiff** — Ищет различия в двух ppm файлах и создает картинку с этими отличиями

Теперь осталось не много, переходим в папку нашего проекта и генерируем с помощью schhist2web файл index.html который затем открываем в браузере.

?

```
1  cd our_prj
2  mkdir output
3  schhist2web -S our_prj.sch ./output
```

В папке output вы найдете необходимый файл index.html , запустите в браузере и получиться что-то вроде вот этого.

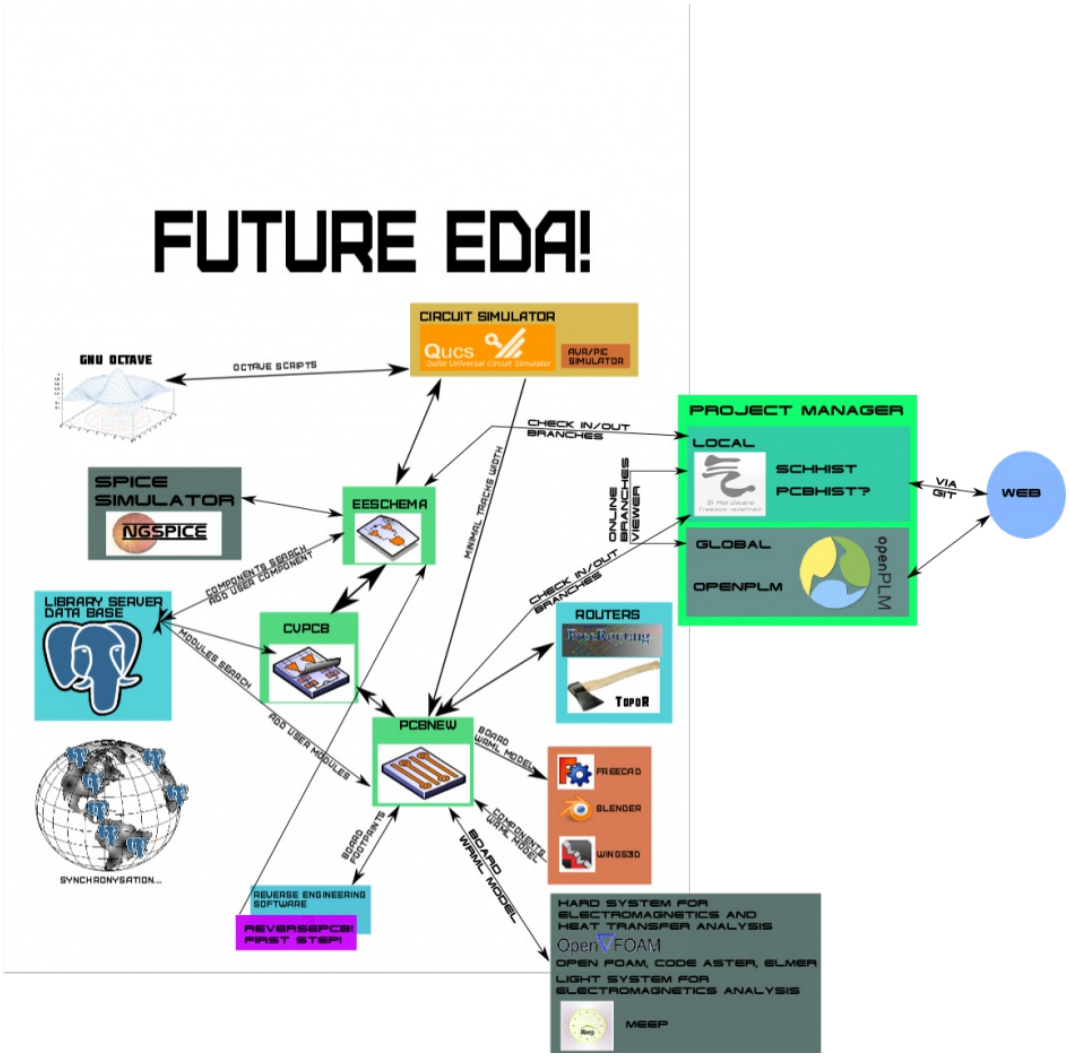
Это простая и базовая инструкция у программы schhist2web есть множество опций,но пока описывать их не будем потому что…

Что делать дальше?

И знаете , что Мы можем сделать и сделаем Мы написали людям из Qi Hardware и они готовы к переносу schhist и написанию утилиты pcbhist(тоже но с файлами плат), Мы создали официальный blueprint о включении кода от Qi Hardware, в официальные выпуски KiCAD. Теперь дело за Вами, если Вам нужна это программа Вы можете написать сюда или в рассылку разработчиков, Вы можете самостоятельно портировать , а также Вы можете спонсировать разработчиков.

Вы идеале если бы kicad поддерживала плагины, то необходимо бы было добавить кнопку, которая вместо сохранения делает коммит нужных файлов, а также управление ветками и прочее, естественно в виде плагина т.к. помимо git существуют mercurial и bazaar.

И в конце хотел бы добавить, небольшую картинку с моим предположением направлений развития kicad


FUTURE EDA!
CIRCUIT SIMULATOR
Qucs
AVR/PIC SIMULATOR
GNU OCTAVE
OCTAVE SCRIPTS
PROJECT MANAGER
LOCAL
SCHHIST
PCBHIST?
GLOBAL
OPENPLM
openPLM
VIA GIT
WEB
CHECK IN/OUT BRANCHES
ONLINE BRANCHES VIEWER
MINIMAL TRACKS WIDTH
SPICE SIMULATOR
NGSPICE
EESCHEMA
COMPONENTS SEARCH
ADD USER COMPONENT
LIBRARY SERVER DATA BASE
CVPCB
MODULES SEARCH
ADD USER MODULES
ROUTERS
FreeRouting
TopoR
PCBNEW
BOARD VRML MODEL
FREECAD
BLENDER
WINGS3D
COMPONENTS VRML MODEL
BOARD VRML MODEL
BOARD FOOTPRINTS
SYNCHRONYSATION...
REVERSE ENGINEERING SOFTWARE
REVERSEPCB! FIRST STEP!
HARD SYSTEM FOR ELECTROMAGNETICS AND HEAT TRANSFER ANALYSIS
OpenFOAM
OPEN FOAM, CODE ASTER, ELMER
LIGHT SYSTEM FOR ELECTROMAGNETICS ANALYSIS
MEEP