



Dateiverwaltung & Versionskontrolle mit Git

Einführung



Inh. Dipl. Ing. Mario Blunk

Buchfinkenweg 3
99097 Erfurt / Deutschland

Telefon 0176 2904 5844

Email info@blunk-electronic.de

Internet www.blunk-electronic.de

Ihre Dateiverwaltung

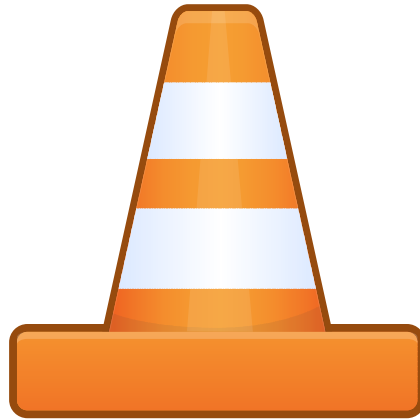


Gegenwärtiger Zustand	Sollzustand
Arbeitsstände in VIELEN Verzeichnissen gespeichert	nur EIN Projektverzeichnis
Arbeitsstand/Version zum Zeitpunkt T ist NICHT einsehbar.	Arbeitsstand/Version zum Zeitpunkt T und dessen Autor JEDERZEIT einsehbar.
Änderungen lassen sich NICHT oder nur sehr UMSTÄNDLICH nachvollziehen.	Änderungen sind in KLARTEXT in Projekthistorie nachvollziehbar.
Änderungen lassen sich NUR mit PROPRIETÄREN Werkzeugen VERSCHIEDENER SW-Applikationen verfolgen.	Änderungen sind mit nur EINEM Werkzeug in ALLEN SW-Applikationen einsehbar.
BINDUNG an SW-Hersteller	KEINE Bindung an SW-Hersteller
Datenhaltung zentral auf EINEM Server	Daten REDUNDANT verteilt auf Rechner der Mitarbeiter.
Mitarbeiter können NUR on-line arbeiten.	Mitarbeiter könne on-line UND off-line arbeiten.
Lizenz, Gebühren, Bindung an Hersteller des Werkzeuges für Dateiverwaltung	KEINE Lizenzen oder Gebühren, KEINE Bindung weil Quellcode offen



Überblick

1. Bestandsaufnahme
2. Was ist Versionskontrolle ?
3. Warum brauchen wir Versionskontrolle ?
4. Was ist Git und was bringt Git ?
5. Beispiele
6. Einschränkungen
7. Fazit

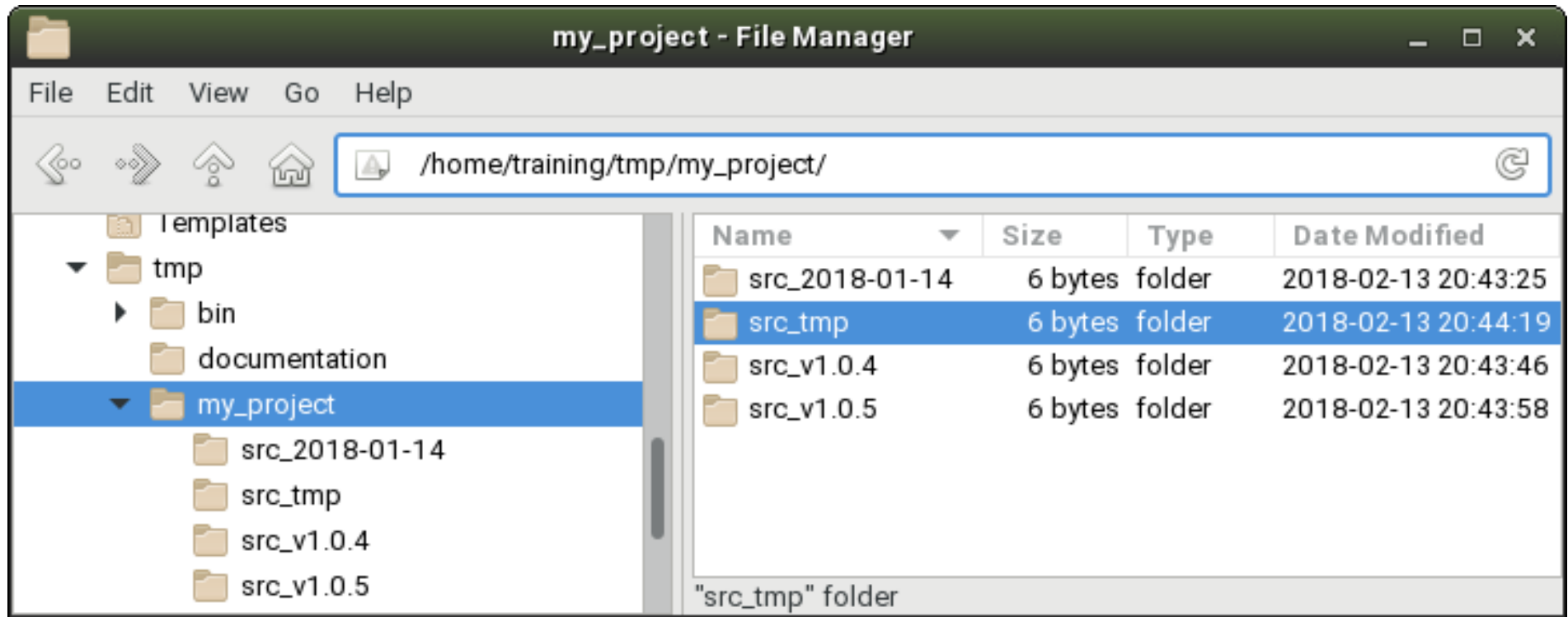


Hinweise für die Anwender am "low end", also Hobbyisten: Einzelkämpfer, kleine bis mittlere Komplexität eines Projekts: Ab wann lohnt es sich, GIT einzusetzen.

Bestandsaufnahme #1



Wie verwalte ich zur Zeit Dateien ?



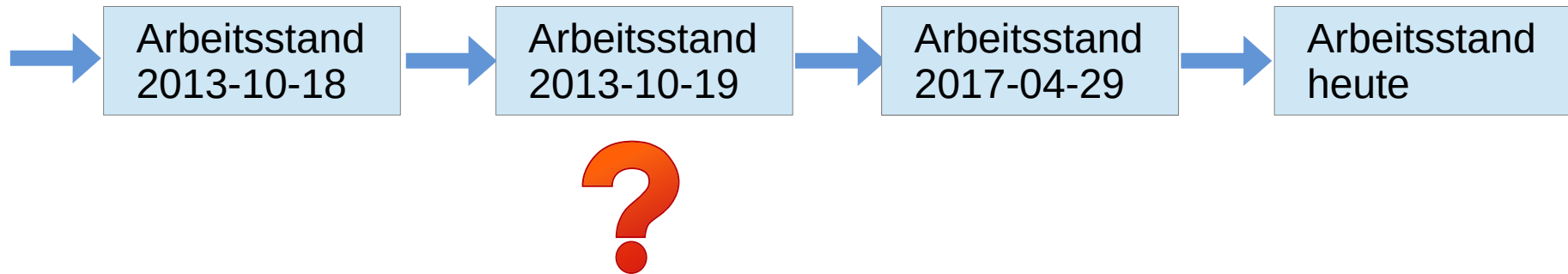
Nachteile:

- Verfolgen von Änderungen **äußerst umständlich**
- Archivierung **bedingt** zuverlässig

Bestandsaufnahme #2



Kann ich den Arbeitsstand zum Zeitpunkt T einsehen ?

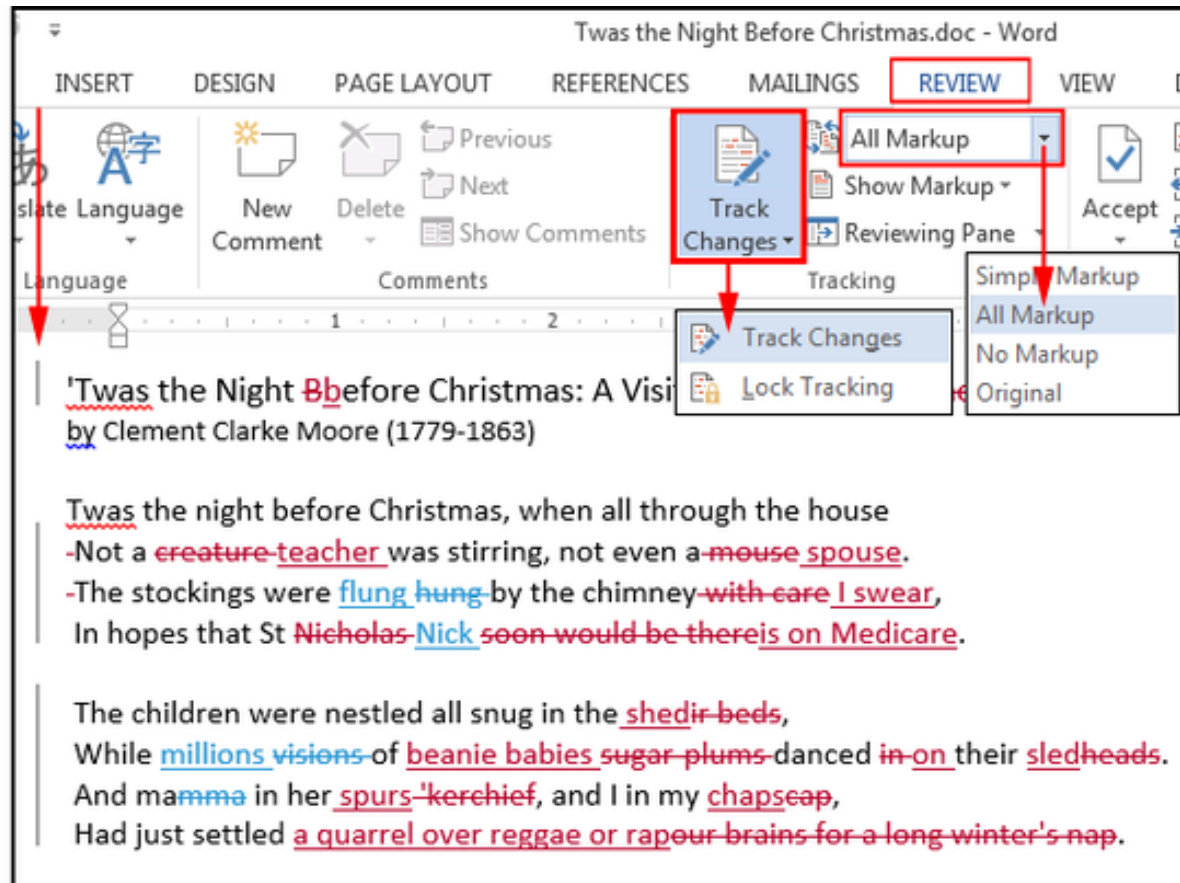


- Wer hat zum Zeitpunkt T was gemacht ?

Bestandsaufnahme #3a



Kann ich Änderungen unabhängig von der Applikation verfolgen ?



Bestandsaufnahme #3b



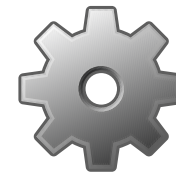
Kann ich Änderungen unabhängig von der Applikation verfolgen ?

Projekt "Fahrrad"



Dokumentation

- MS Office
- LibreOffice
- ...



Mechanik-Konstruktion

- Solid Works
- Autocad
- FreeCad
- ...

```
ld> BC,OFFSET
ld> A,B
ld> I,A>;load
im> 2> ;enabl
```

Software

- GCC
- MPLAB
- STM32 IDEs
- GPS (GNAT)



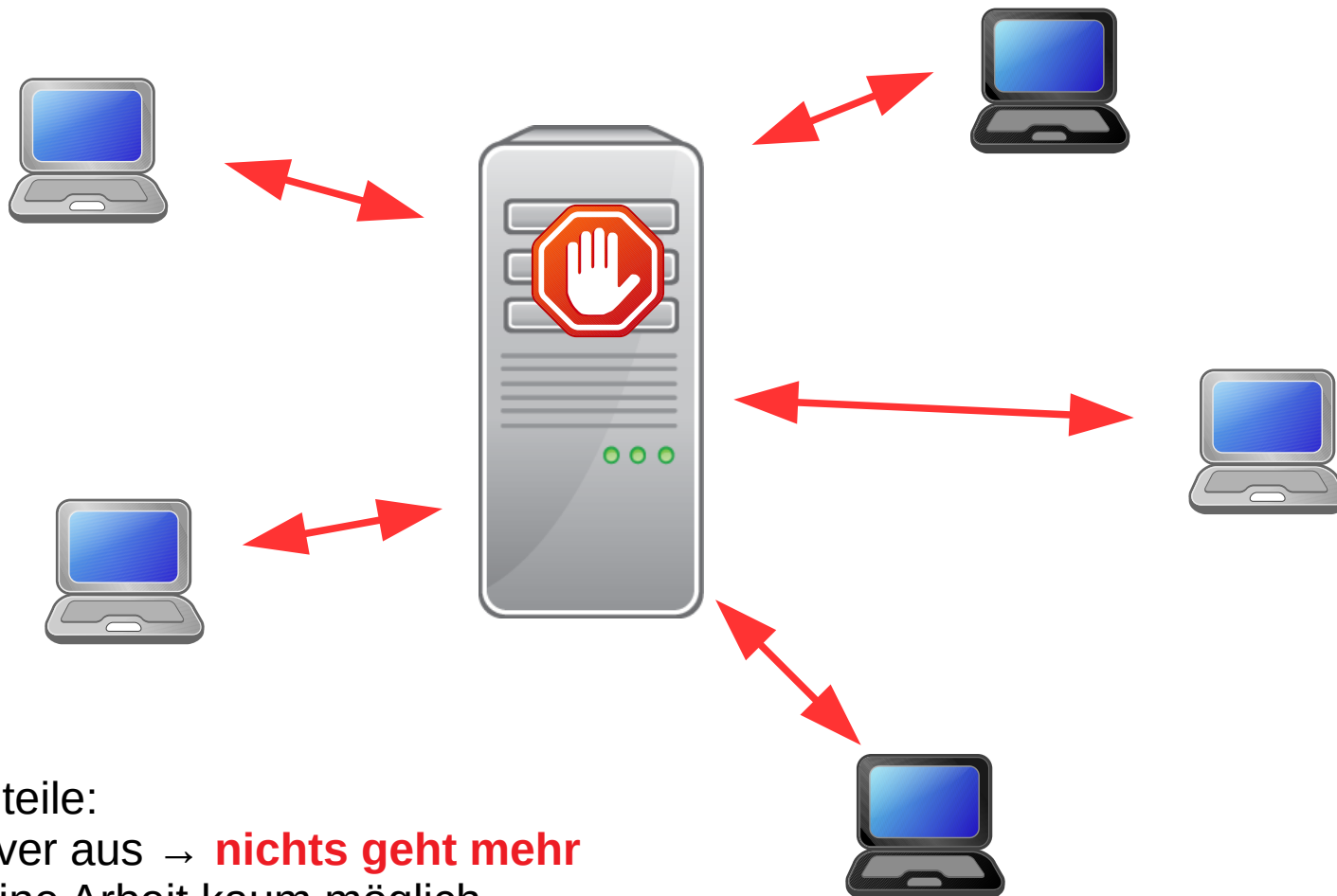
Elektronik-HW

- KiCad
- EAGLE
- Altium
- Vivado
- ...

Bestandsaufnahme #4



Sind wir auf einen zentralen Server angewiesen ?



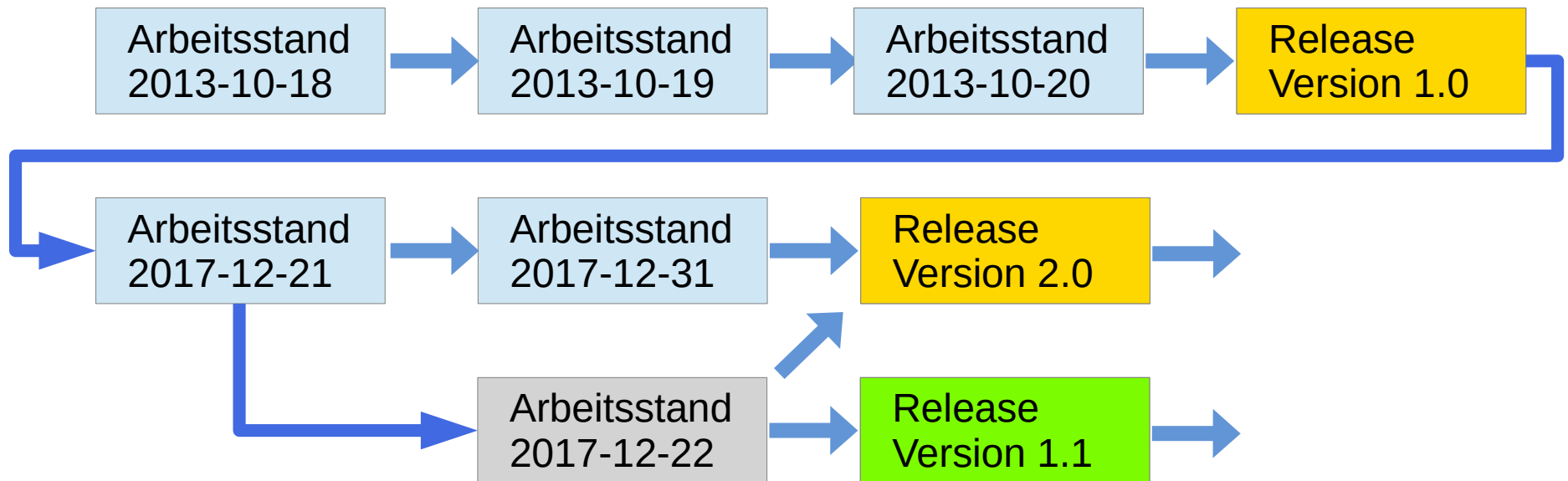
Nachteile:

- Server aus → **nichts geht mehr**
- off-line Arbeit kaum möglich

Was ist Versionskontrolle ?

Es geht um Quellcode und Klartext (ASCII, XML, ...)

1. Archivieren
2. Verwalten
3. Verfolgen von Änderungen
4. Release/Versionen (wie z.B. V2.16.1, ...)

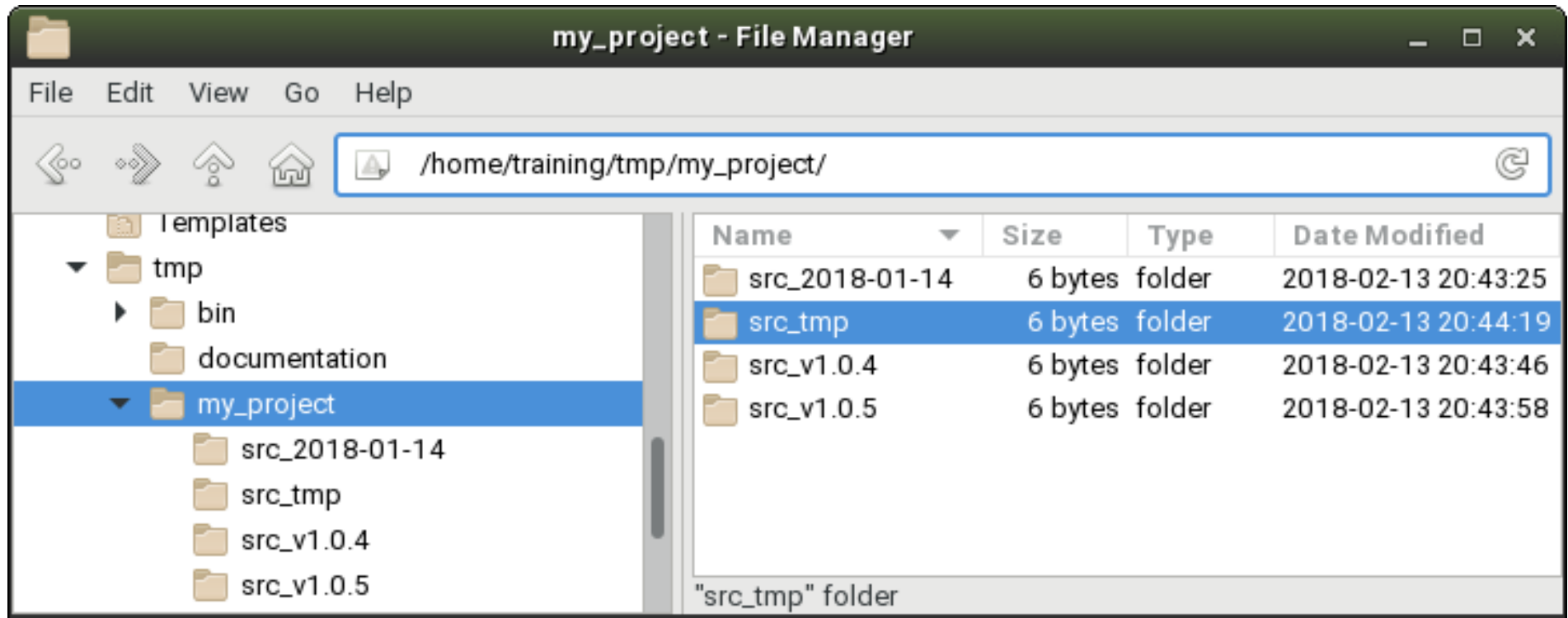


Warum Versionskontrolle



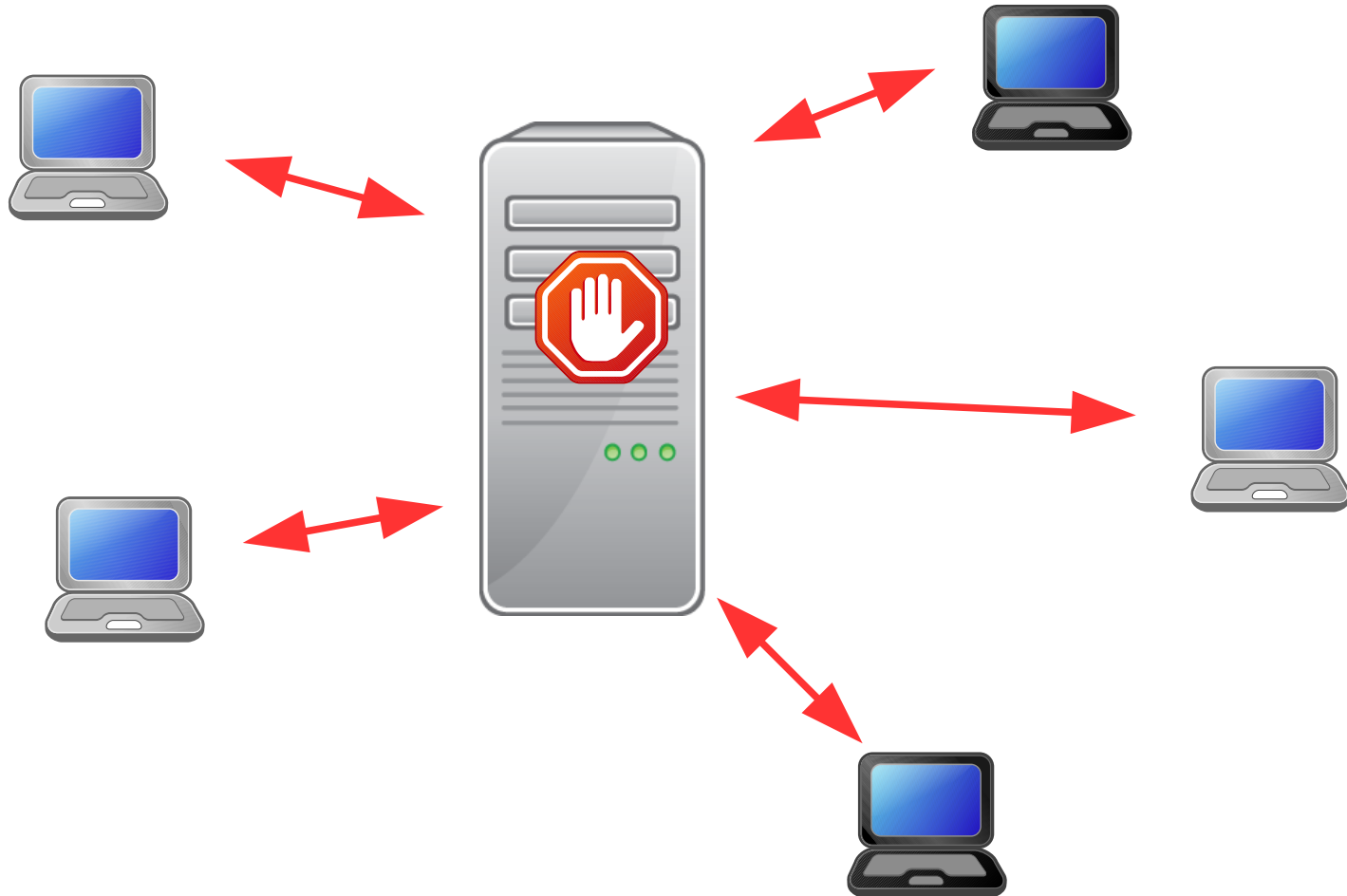
1. erleichterte Fehlersuche/Debugging
2. Archivierung von Konstruktionsdaten (CAD), Quellcode, Dokumentation, Fertigungsdaten (CAM) ...
3. Rückverfolgung von Änderungen
4. für Zertifizierungen (ISO 9001, IEC 61508 / 61511 / 62061, DO-178B, MIL-STD-882-E, ...)

Verzeichnisbasiert



So nicht !

Zentraler Server

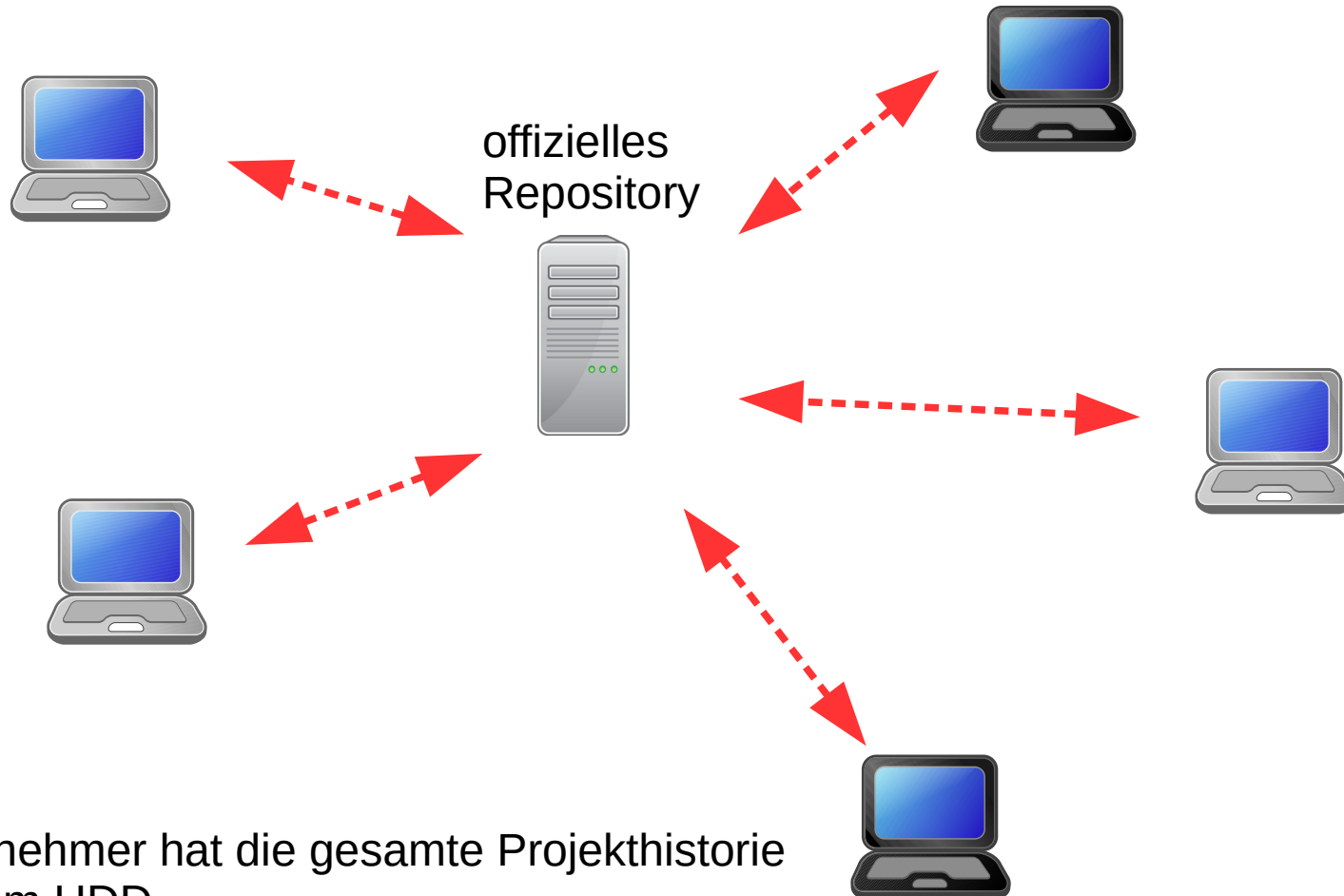


Nachteile:

- Server aus → **nichts geht mehr**
- off-line Arbeit kaum möglich



Alles ist lokal - Git



Vorteile:

- Jeder Teilnehmer hat die gesamte Projekthistorie auf lokalem HDD
- off-line Arbeit möglich
- Datensicherheit
- OpenSource (OSS)

Beispiel #1a Assembler

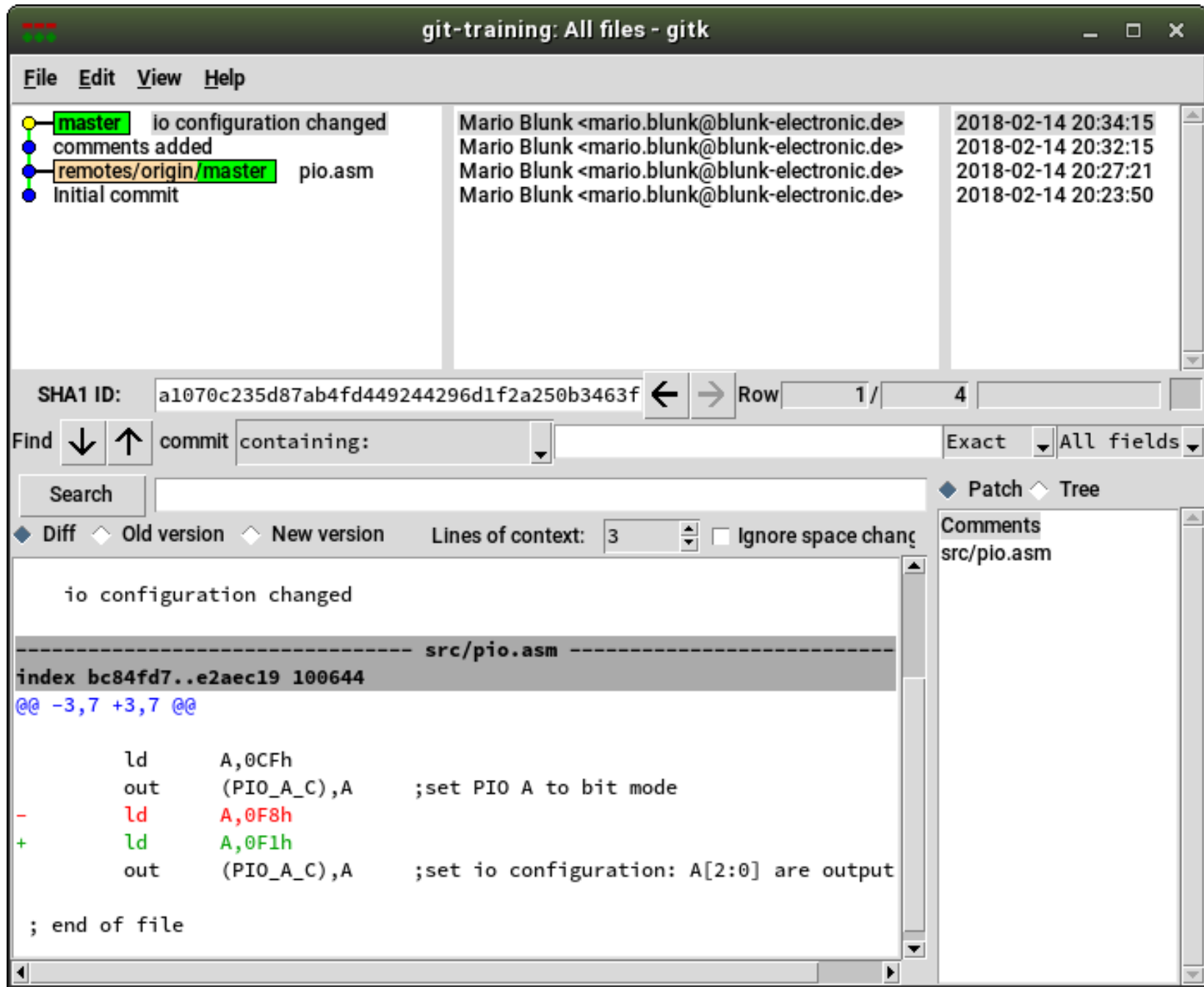
ursprünglicher Code

```
ld    A,0CFh
out   (PIO_A_C),A    ;set PIO A to bit mode
ld    A,0F8h
out   (PIO_A_C),A    ;set io configuration: A[2:0] are outputs
```

geänderter Code

```
ld    A,0CFh
out   (PIO_A_C),A    ;set PIO A to bit mode
ld    A,0F1h
out   (PIO_A_C),A    ;set io configuration: A[2:0] are outputs
```


Beispiel #1b Assembler



The screenshot shows a gitk window titled "git-training: All files - gitk". The top panel displays the commit history for the "pio.asm" file. The commit list shows four commits, all by "Mario Blunk <mario.blunk@blunk-electronic.de>". The top commit is highlighted in green and labeled "master", with the message "io configuration changed". Below it are three commits labeled "remotes/origin/master", with messages "comments added", "pio.asm", and "Initial commit".

The middle panel shows the SHA1 ID: a1070c235d87ab4fd449244296d1f2a250b3463f. Below this is a search bar with "commit containing:" and a search button. The bottom panel shows a diff view for "src/pio.asm". The diff is in "Patch" mode, showing a change in the file. The diff header is "index bc84fd7..e2aec19 100644". The diff content shows a change in the "ld" instruction for the "pio.asm" file. The original code (red) is "ld A,0F8h" and the new code (green) is "ld A,0F1h". The comment for the new code is ";set io configuration: A[2:0] are output".

```
io configuration changed

----- src/pio.asm -----
index bc84fd7..e2aec19 100644
@@ -3,7 +3,7 @@
     ld    A,0CFh
     out   (PIO_A_C),A      ;set PIO A to bit mode
-    ld    A,0F8h
+    ld    A,0F1h
     out   (PIO_A_C),A      ;set io configuration: A[2:0] are output

; end of file
```



Beispiel #1c Assembler

letzten Stand vom offiziellen Repo holen
\$ **git pull**

Arbeiten: zeichnen, programmieren, dokumentieren

gegenwärtigen Dateiinhalt auf Index setzen
\$ **git add pio.asm**

Änderung einchecken
\$ **git commit -m "io configuration geändert"**

neuen Stand zum offiziellen Repo senden
\$ **git push**



auch mit GUI verfügbar (Linux, Windows, Mac, ...)

Beispiel #2a Hochsprache C

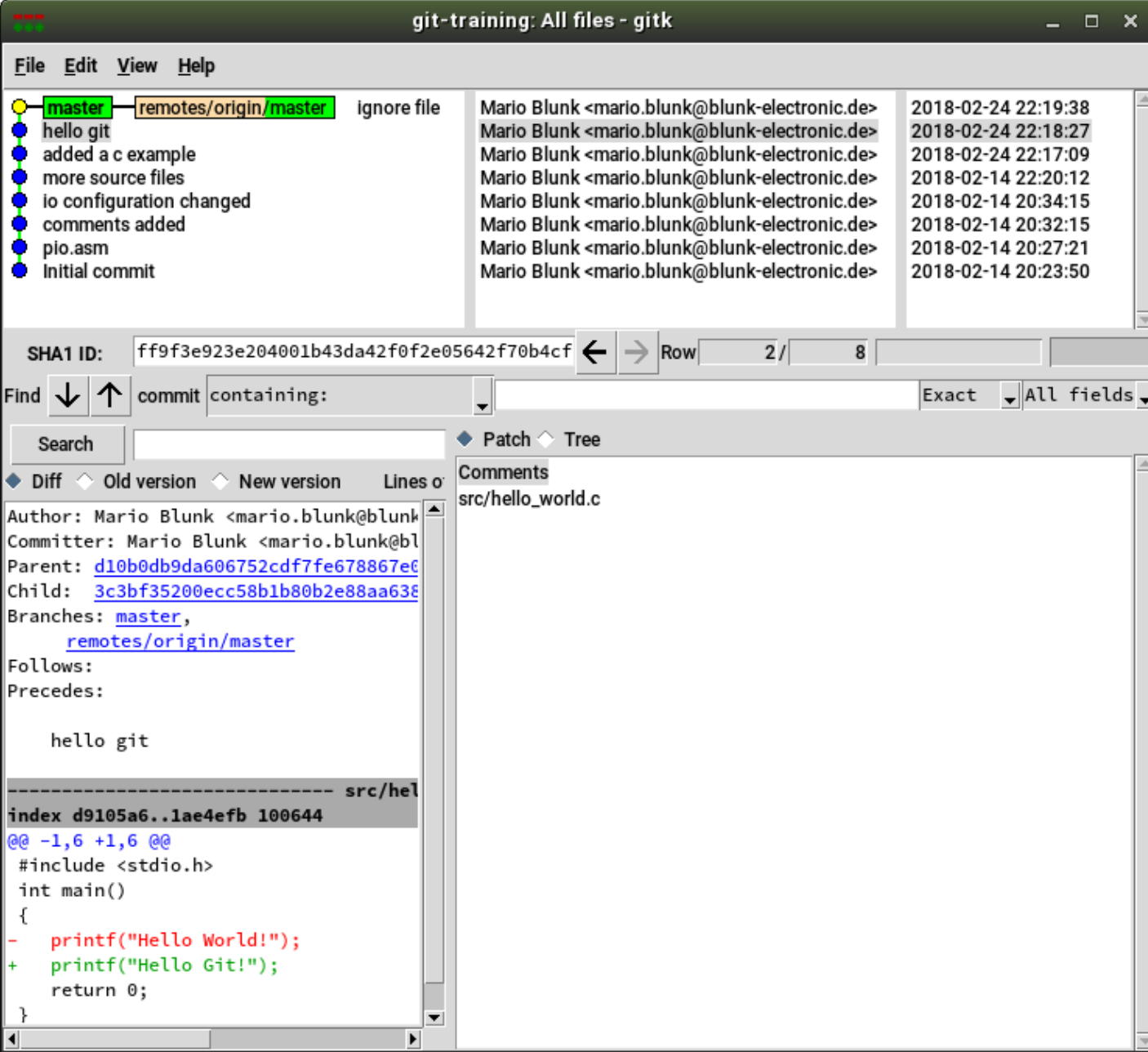
ursprünglicher Code

```
#include <stdio.h>
int main()
{
    printf("Hello World!");
    return 0;
}
```

geänderter Code

```
#include <stdio.h>
int main()
{
    printf("Hello Git!");
    return 0;
}
```

Beispiel #2b Hochsprache C



The screenshot shows a Git GUI window titled "git-training: All files - gitk". The window displays a commit history on the left, a commit details view in the middle, and a diff view on the right. A red arrow points to the diff view.

Commit History:

- master — remotes/origin/master ignore file
- hello git
- added a c example
- more source files
- io configuration changed
- comments added
- pio.asm
- Initial commit

Commit Details:

SHA1 ID: ff9f3e923e204001b43da42f0f2e05642f70b4cf Row 2 / 8

Find ↓ ↑ commit containing: Exact All fields

Search

◆ Patch ◆ Tree

◆ Diff ◆ Old version ◆ New version Lines of

Comments
src/hello_world.c

Author: Mario Blunk <mario.blunk@blunk-
Committer: Mario Blunk <mario.blunk@bl
Parent: [d10b0db9da606752cdf7fe678867e6](#)
Child: [3c3bf35200ecc58b1b80b2e88aa638](#)
Branches: [master](#),
[remotes/origin/master](#)
Follows:
Precedes:

hello git

----- src/hel

index d9105a6..1ae4efb 100644

```
@@ -1,6 +1,6 @@
#include <stdio.h>
int main()
{
- printf("Hello World!");
+ printf("Hello Git!");
return 0;
}
```

Beispiel #3 Verilog HDL

ursprünglicher Code

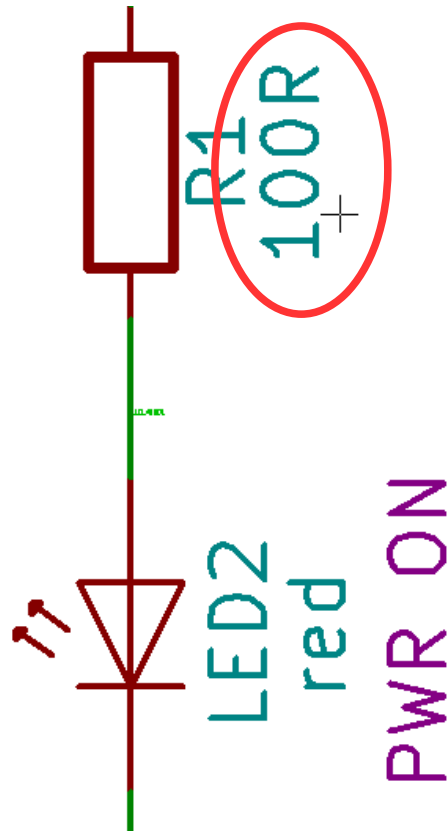
```
module i2c_master(  
    clk,          // input  
    reset,       // input synchronous  
    sda,         // inout  
    scl,         // inout
```

geänderter Code

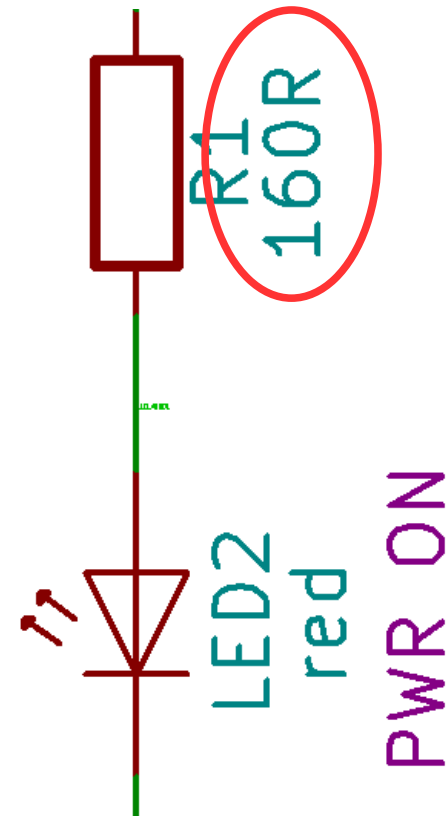
```
module i2c_master(  
    clk,          // input  
    reset_n,     // input asynchronous  
    reset,       // input synchronous  
    sda,         // inout  
    scl,         // inout
```

Beispiel #4a KiCad

ursprünglicher Schaltplan

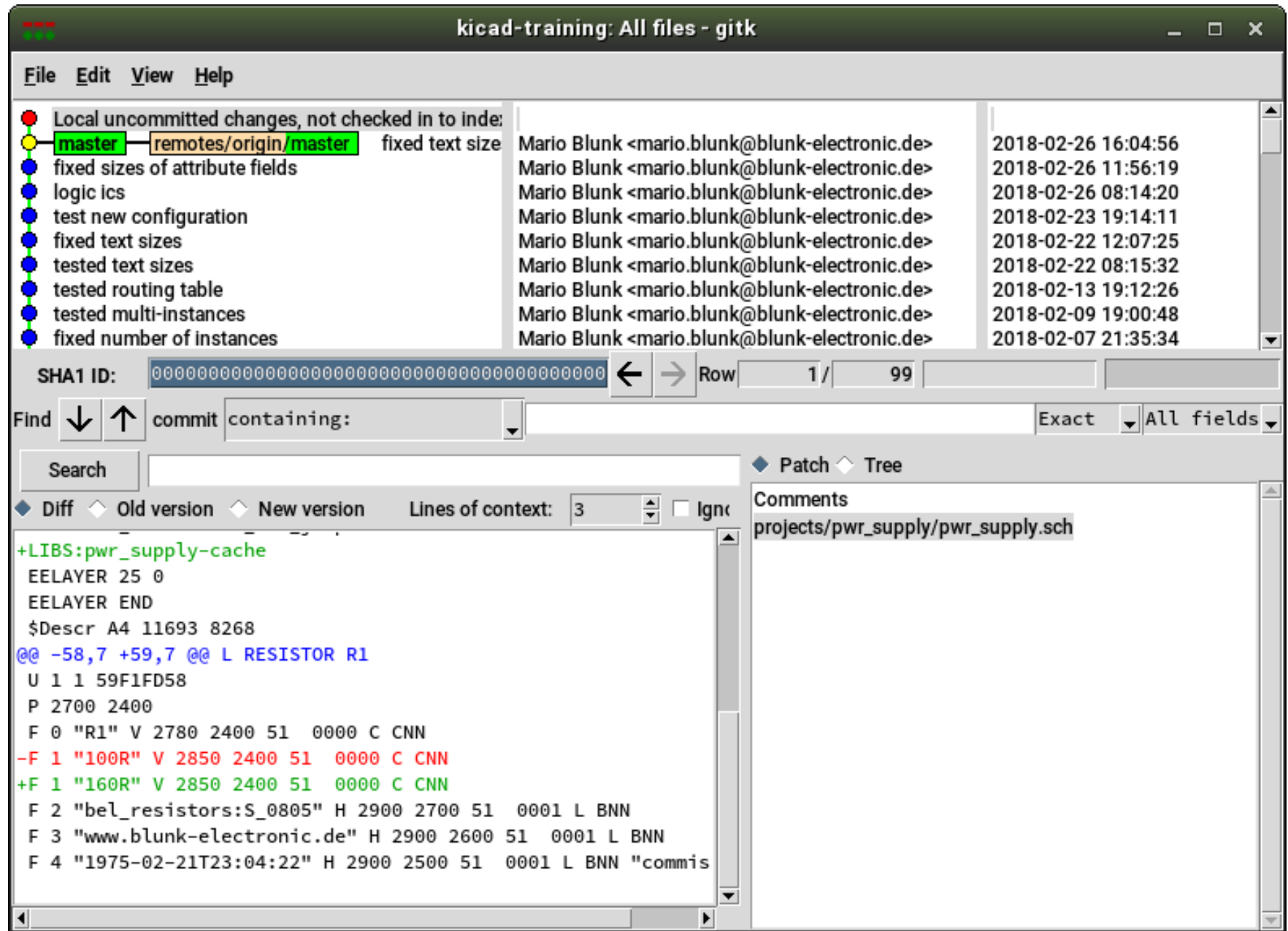


geänderter Schaltplan



auch für Materiallisten (in *.csv) verwendbar

Beispiel #4b KiCad



The screenshot shows the gitk interface for a repository named "kicad-training". The top panel displays a commit history with the following entries:

Commit Message	Author	Date
Local uncommitted changes, not checked in to index		
fixed text size	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-26 16:04:56
fixed sizes of attribute fields	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-26 11:56:19
logic ics	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-26 08:14:20
test new configuration	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-23 19:14:11
fixed text sizes	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-22 12:07:25
tested text sizes	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-22 08:15:32
tested routing table	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-13 19:12:26
tested multi-instances	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-09 19:00:48
fixed number of instances	Mario Blunk <mario.blunk@blunk-electronic.de>	2018-02-07 21:35:34

The bottom panel shows a diff view of a commit. The diff highlights changes to the file "projects/pwr_supply/pwr_supply.sch". The changes include:

- Adding a library cache: `+LIBS:pwr_supply-cache`
- Adding a resistor component: `+F 1 "160R" V 2850 2400 51 0000 C CNN`
- Removing a resistor component: `-F 1 "100R" V 2850 2400 51 0000 C CNN`
- Adding component footprints: `F 2 "bel_resistors:S_0805" H 2900 2700 51 0001 L BNN`, `F 3 "www.blunk-electronic.de" H 2900 2600 51 0001 L BNN`, and `F 4 "1975-02-21T23:04:22" H 2900 2500 51 0001 L BNN "commis"`

A red arrow points to the diff view.

Einschränkungen

Es geht um Quellcode und Klartext (ASCII, XML, ...)

Änderungsverfolgung nicht möglich mit:

*.doc / *.docx

*.ppt / *.pptx

*.odt / *.ods (statt dessen *.fodt / *.fodp verwenden)

*.schdoc

*.pcbdoc

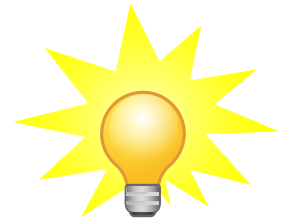
*.bin

*.jpg / *.bmp ...

*.wav / *.mp3 / *.ogg

.....

Fazit



1. Archivieren
2. Verwalten
3. Verfolgen von Änderungen → vereinfachte Fehlersuche
4. wenn OpenSource Werkzeuge verwendet werden → keine Bindung an SW-Hersteller
5. dezentrale Datenhaltung → Redundanz → Datensicherheit
6. Projektteilnehmer können off-line arbeiten
7. kostenfreies Werkzeug zur Versionskontrolle



Links

http://www.blunk-electronic.de/pdf/git_training_teil_1.pdf



<https://de.wikipedia.org/wiki/Git>

<https://de.wikipedia.org/wiki/GitHub>

<https://de.wikipedia.org/wiki/GitLab>

<https://de.wikipedia.org/wiki/Bitbucket>

<https://git-scm.com/doc>

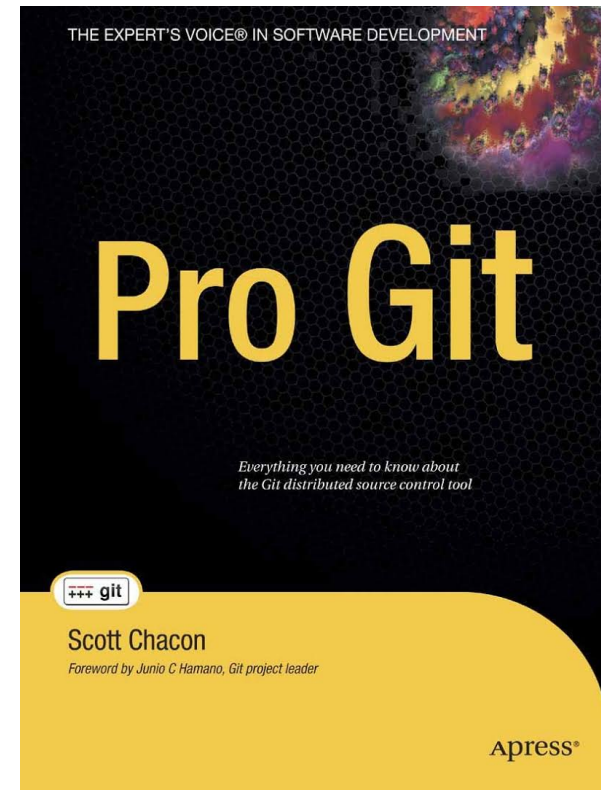
<https://github.com/Blunk-electronic>

Literatur

<http://gitbu.ch/>



<https://git-scm.com/book/de/v1>
<https://git-scm.com/book/en/v2>



<https://progit2.s3.amazonaws.com/en/2016-03-22-f3531/progit-en.1084.pdf>

Danke für Ihre Aufmerksamkeit !