

# Automatisierte Build- und Runtime-Tests mit tbot

Heiko Schocher <[hs@denx.de](mailto:hs@denx.de)>

29.11.2016

# Heiko Schocher

- Software engineer at DENX S.E. since 2004
- U-Boot Custodian
- Hacking U-Boot/Linux

# Inhalt

- Kurze Geschichte von tbot
- Grundlegende tbot Prinzipien
- Eventsystem
- Installation
- Beispiele
- links
- Fragen

# Geschichte

- 2012 start der Entwicklung,  
mit dem Ziel linux Kommandos zu automatisieren
- DUTS Ideen:  
<http://www.denx.de/wiki/DUTS/DUTSDocs>
- Python
- Ziel: Kommandlinetool, das waehrend der Entwicklung schon genutzt werden kann.

# Grundlegende Prinzipien

## Was ist tbot ?

- commandline tool
- öffnet/arbeitet auf ssh Verbindungen
- startet shell Kommandos über diese Verbindungen
- auswerten der Ausgaben der shell Kommandos

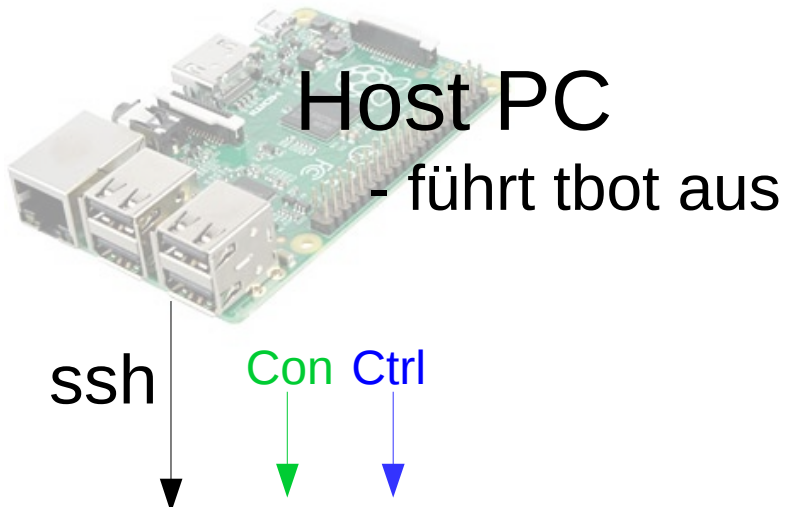
# grundlegende Prinzipien Testcase TC

- ➔ Was ist ein Testcase ?
  - python code der:
  - shell Kommandos über die ssh Verbindungen verschickt
  - die Ausgaben dieser Kommandos auswertet
  - und daraufhin entscheidet, ob der TC erfolgreich oder erfolglos ist
- ➔ TC kann andere TC aufrufen
- ➔ Tbot sucht in src/tc nach den Testcases
- ➔ TC definiert für welchen boardstate er gueltig ist

# grundlegende Prinzipien Board State

- Board State entspricht der Software, die getestet werden soll.
- Der Board State wird durch das shell prompt erkannt.
- Bevor shell Kommands an das Board gesendet werden, versucht tbot in den Board State zu gelangen. Schlägt dies fehl, so gilt der TC als erfolglos.
- Im Moment werden 2 Board States unterstützt:  
„U-Boot“ und „linux“

# grundlegende Prinzipien Testsetup



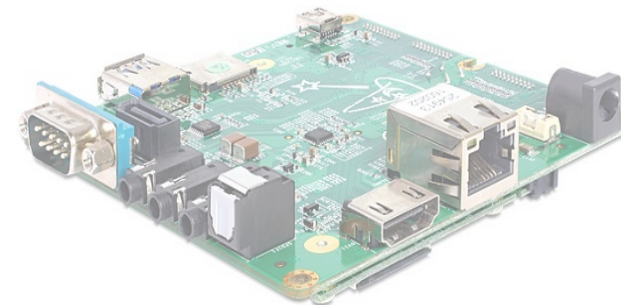
## Virtuelles Labor VL

### Lab PC

- connect to board(s) console
- power on/off board(s)
- read current power state
- nfs/tftp server
- get and compile sources
- ...



### Board



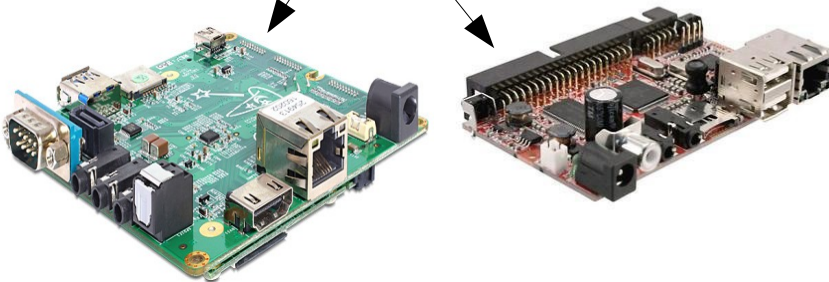


# Mein „Traum“



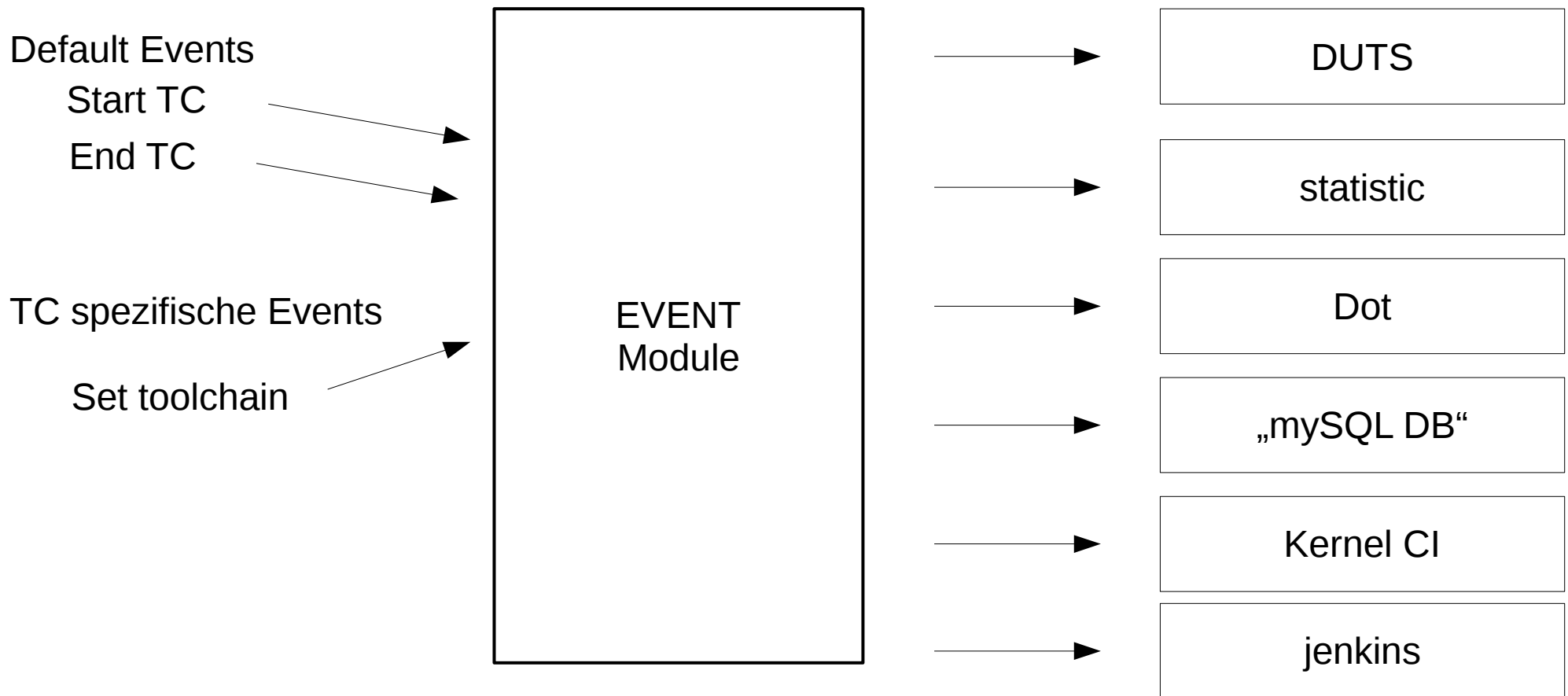
Tbot Host

[...]



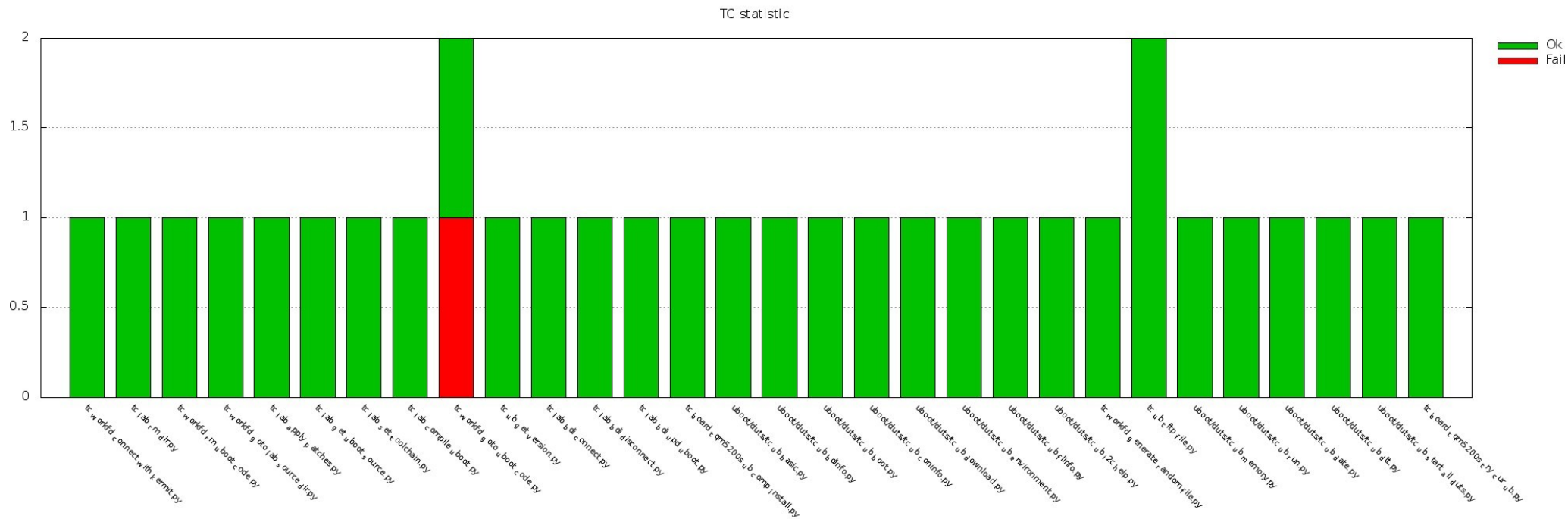
# Tbot Events

Während der Ausführung von tbot, werden Events erzeugt, die im Event Module gesammelt werden:



# Event backends Beispiele

[https://github.com/hsdenx/tbot/blob/master/src/common/event/statistic\\_plot.py](https://github.com/hsdenx/tbot/blob/master/src/common/event/statistic_plot.py)



[http://xeidos.ddns.net/tbot/id\\_69/statistic.jpg](http://xeidos.ddns.net/tbot/id_69/statistic.jpg)

# Event backends Beispiele

<https://github.com/hsdenx/tbot/blob/master/src/common/event/dot.py>



[http://xeidos.ddns.net/tbot/id\\_62/graph.png](http://xeidos.ddns.net/tbot/id_62/graph.png)

# Tbot Installation

# Tbot Installation

## Übersicht

- Tbot code herunterladen:  
git clone <https://github.com/hsdenx/tbot.git>
- Tbot nutzt das python modul paramiko  
<http://www.paramiko.org/installing.html>
- Verzeichnis für log dateien anlegen
- Passwort Datei erzeugen
- Board Konfigurationsdatei erzeugen
- Labor Konfigurationsdatei erzeugen
- Einmalig: virtuelles Labor einbinden
- TC schreiben

# Tbot Installation

## Verzeichnis für Logdateien

- Ganz einfach, Sie müssen nur ein Verzeichnis erstellen, indem Sie die tbot Logdateien speichern wollen.
- Übergeben dieses Verzeichnisses beim start mittels der Kommandline option "-l".
- Default: „log“ in den tbot sourcen
- Alle Zeichen die gesendet oder empfangen werden, werden dort gespeichert.
- Durch verschiedene „log levels“ können noch andere Informationen ein/aus geschaltet werden.

# Tbot Installation

## tbot password Konfiguration

- tbot braucht Passwörter für:
  - Verbindungsaufnahme zum VL
  - manche TC brauchen root zugriff
  - login zu board console

Die Passwörter sind in dem python file password.py gespeichert, welches tbot beim starten einliest.



# Tbot Installation

## Password Konfiguration

```
# passwords for the lab
if (board == 'lab'):
    if (user == 'hs'):
        password = 'XXXX'
    if (user == 'root'):
        password = 'XXXX'
...
elif (board == 'smartweb'):
    if (user == 'root'):
        password = 'XXXX'
...
```

# Tbot Installation

## Boardkonfiguration

- tbot liest eine boardspezifische Konfigurationsdatei beim Start ein.
- Diese wird beim tbot Start durch die Kommandline option „-c“ übergeben
- Inhalt:
  - board spezifische Angaben, z.B: U-Boot/Linux prompt
  - board spezifische TC variablen

# Tbot Boardkonfigdatei: Beispiel

```
# tbot configuration
# for the smartweb board
boardname = 'smartweb'
tftpboardname = 'smartweb_hw'

uboot_prompt = 'U-Boot# '
linux_prompt = 'ttbott> '

create_dot = 'yes'
create_statistic = 'yes'
...

# variables used in testcases
ub_load_board_env_addr = '0x21000000'
tc_lab_get_uboot_source_git_repo = "/home/git/u-boot.git"
tc_lab_get_uboot_source_git_branch = "master"
```

<https://github.com/hsdenx/tbot/blob/master/config/smartweb.py>

# Tbot Installation

## VL Konfigurationsdatei

- Tbot liest beim starten eine Konfigurationsdatei für das VL ein.
- Diese wird beim start durch die Kommandline option „-s“ übergeben.
- Inhalt:
  - wo findet tbot das VL
  - Arbeitsverzeichnisse
  - ...

# Tbot VL Konfigdatei: Beispiel

[https://github.com/hsdenx/tbot/blob/master/config/lab\\_denx.py](https://github.com/hsdenx/tbot/blob/master/config/lab_denx.py)

```
# tbot configuration
# for the denx vlab
ip = 'pollux.denx.org'
user = 'hs'
accept_all = True
keepalivetimeout = 1
channel_timeout = 0.5
loglevel = 'INFO'
wdt_timeout = '900'

tc_workfd_work_dir = '/work/hs/tbot'
lab_tmp_dir = '/var/tmp/'
```

# Tbot Installation

## VL Integration

- Erstellen eines TC für Ein/Aus der Boardpower
- Erstellen eines TC der den aktuellen Powerzustand des Boards ausliest.
- Erstellen eines TC, der zu der console des Boards verbindet.
- Um diese TC zu verwenden müssen diese nun in der Laborkonfiguration angegeben werden.

# Tbot VL Integration: Beispiel

[https://github.com/hsdenx/tbot/blob/master/config/flea3\\_home.py#L21](https://github.com/hsdenx/tbot/blob/master/config/flea3_home.py#L21)

```
21 # set connect testcase (as it is with kermit, not with connect)
22 tc_lab_denx_connect_to_board_tc = 'tc_workfd_connect_with_kermit.py'
23
24 tc_lab_denx_power_tc = 'tc_lab_interactive_power.py'
25 tc_lab_denx_get_power_state_tc = 'tc_lab_interactive_get_power_state.py'
```

# Tbot Installation Dokumentation

Mehr detaillierte Informationen zur Installation:

<https://github.com/hsdenx/tbot/blob/master/doc/README.install>



# Beispiel Testcase

# Tbot: Beispiel TC

- Schritt 1:  
U-Boot test auf dem AT 91 basierenden smartweb board  
(clone, compile, install und Aufruf eines simplen  
U-Boot help Kommand TC)
- Schritt 2:  
Hole alle Patches von meiner U-Boots patchwork ToDo liste,  
wobei ein Patch den U-Boot help Kommand TC zum scheitern bringt.
- Schritt 3:  
Automatisches herausfinden mittels „git bisect“, welcher Patch den  
Fehler verursacht.
- Da wir nicht genügend Zeit haben, zeige ich hier nur die Ergebnisse. Ein  
komplettes ungekürztes Video finden Sie hier:

<https://www.youtube.com/watch?v=zfjppj3DLsx4>

# Schritt 1

Quellcode aller TC auf: <https://github.com/hsdenx/tbot>

Hole U-Boot code, compile, installiere diesen auf dem Board und starte den help Kommand U-Boot testcase:

[src/tc/demo/tc\\_demo\\_part1.py](#)

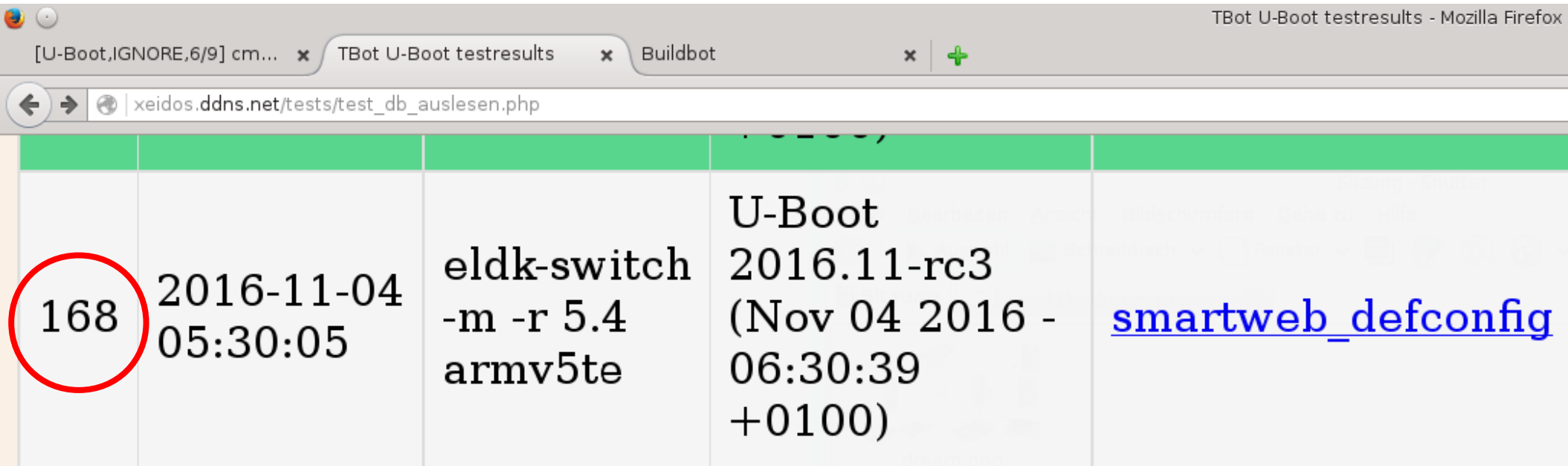
Was macht dieser TC:

- ➔ rufe TC [src/tc/demo/tc\\_demo\\_get\\_ub\\_code.py](#) auf clone mainline U-Boot
- ➔ rufe TC [src/tc/demo/tc\\_demo\\_compile\\_install\\_test.py](#) auf compile und installiere die U-Boot images auf dem Board und rufe den U-Boot help Kommand TC [src/tc/uboot/tc\\_ub\\_help.py](#) auf
- ➔ [src/tc/uboot/tc\\_ub\\_help.py](#) prüft, das der output des help cmd nur 1 „?“ enthält.

# Schritt 1: Vorbereitungen

Webpage@

[http://xeidos.ddns.net/tests/test\\_db\\_auslesen.php](http://xeidos.ddns.net/tests/test_db_auslesen.php)

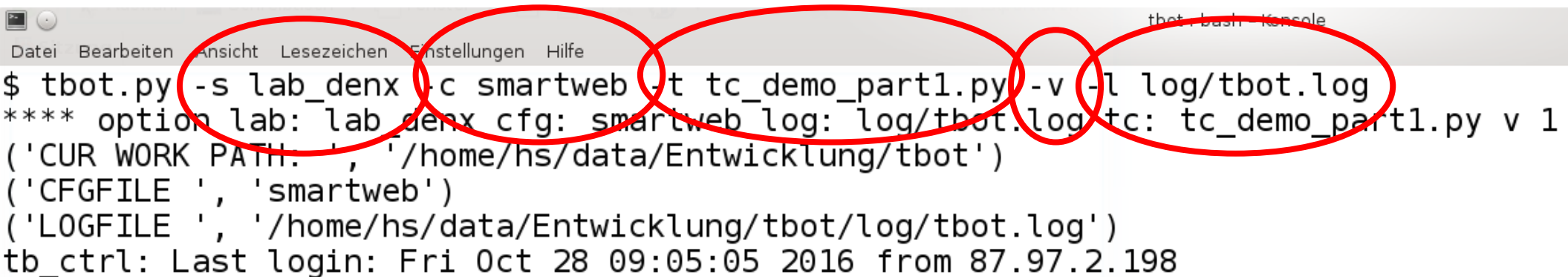


168	2016-11-04 05:30:05	eldk-switch -m -r 5.4 armv5te	U-Boot 2016.11-rc3 (Nov 04 2016 - 06:30:39 +0100)	<a href="#">smartweb_defconfig</a>

Aktuelle ID ist 168, damit muss der nächste Test die ID 169 bekommen ...

# Schritt 1: starten von tbot

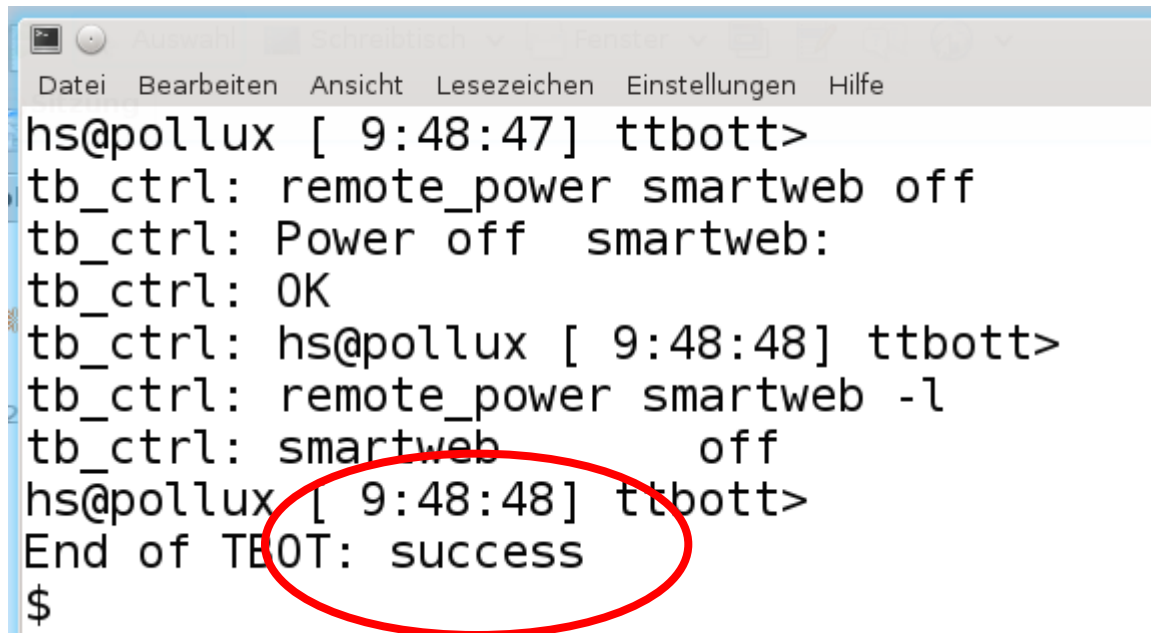
Nun starten wir den Testcase ...



A terminal window titled 'tbot: bash - Konsole' showing the execution of the command `$ tbot.py -s lab_denx -c smartweb -t tc_demo_part1.py -v -l log/tbot.log`. The output displays configuration details and a login message. Red circles are drawn around the arguments `-s lab_denx`, `-c smartweb`, `-t tc_demo_part1.py`, and `-v` in the command line.

```
$ tbot.py -s lab_denx -c smartweb -t tc_demo_part1.py -v -l log/tbot.log
**** option lab: lab_denx cfg: smartweb log: log/tbot.log tc: tc_demo_part1.py v 1
('CUR WORK PATH: ', '/home/hs/data/Entwicklung/tbot')
('CFGFILE ', 'smartweb')
('LOGFILE ', '/home/hs/data/Entwicklung/tbot/log/tbot.log')
tb_ctrl: Last login: Fri Oct 28 09:05:05 2016 from 87.97.2.198
```

# Schritt 1 Ergebnis

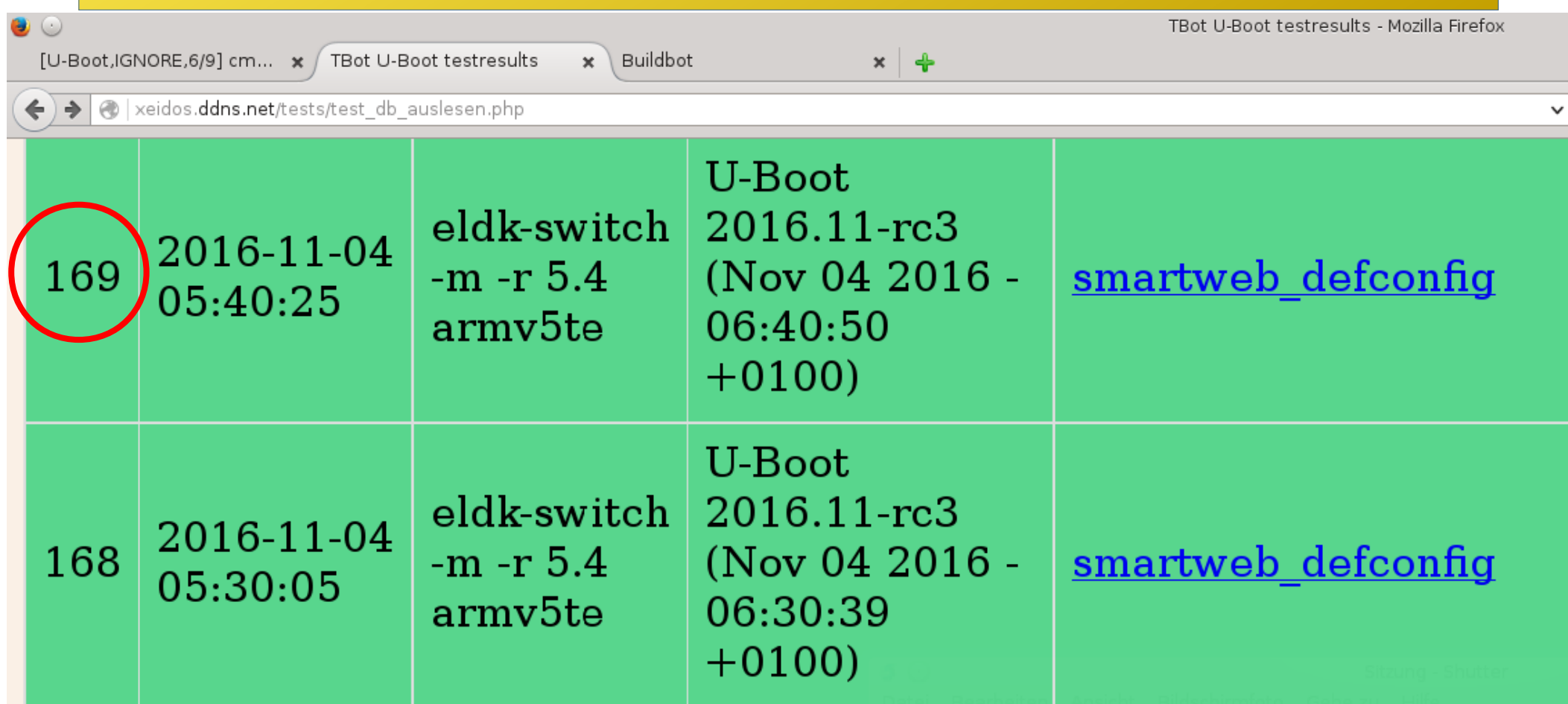


```
hs@pollux [ 9:48:47] ttbott>
tb_ctrl: remote_power smartweb off
tb_ctrl: Power off smartweb:
tb_ctrl: OK
tb_ctrl: hs@pollux [ 9:48:48] ttbott>
tb_ctrl: remote_power smartweb -l
tb_ctrl: smartweb off
hs@pollux [ 9:48:48] ttbott>
End of TBOT: success
$
```

Tbot endet mit success, damit sind alle unsere Testcases mit Erfolg durchgeführt worden

- Schauen wir nun auf unsere Testergebnis webseite
- Da es keine Fehler gab, sollte der Hintergrund grün sein ...

# Schritt 1 Ergebnis



169	2016-11-04 05:40:25	eldk-switch -m -r 5.4 armv5te	U-Boot 2016.11-rc3 (Nov 04 2016 - 06:40:50 +0100)	<a href="#">smartweb_defconfig</a>
168	2016-11-04 05:30:05	eldk-switch -m -r 5.4 armv5te	U-Boot 2016.11-rc3 (Nov 04 2016 - 06:30:39 +0100)	<a href="#">smartweb_defconfig</a>

[http://xegios.ddns.net/tests/test\\_db\\_auslesen.php#169](http://xegios.ddns.net/tests/test_db_auslesen.php#169)

# Schritt 2

Hole alle Patches meiner U-Boot Patchwork ToDo Liste

Checke jeden Patch mit checkpatch.pl und wende die Patches auf dem aktuellen U-Boot source an

TC:

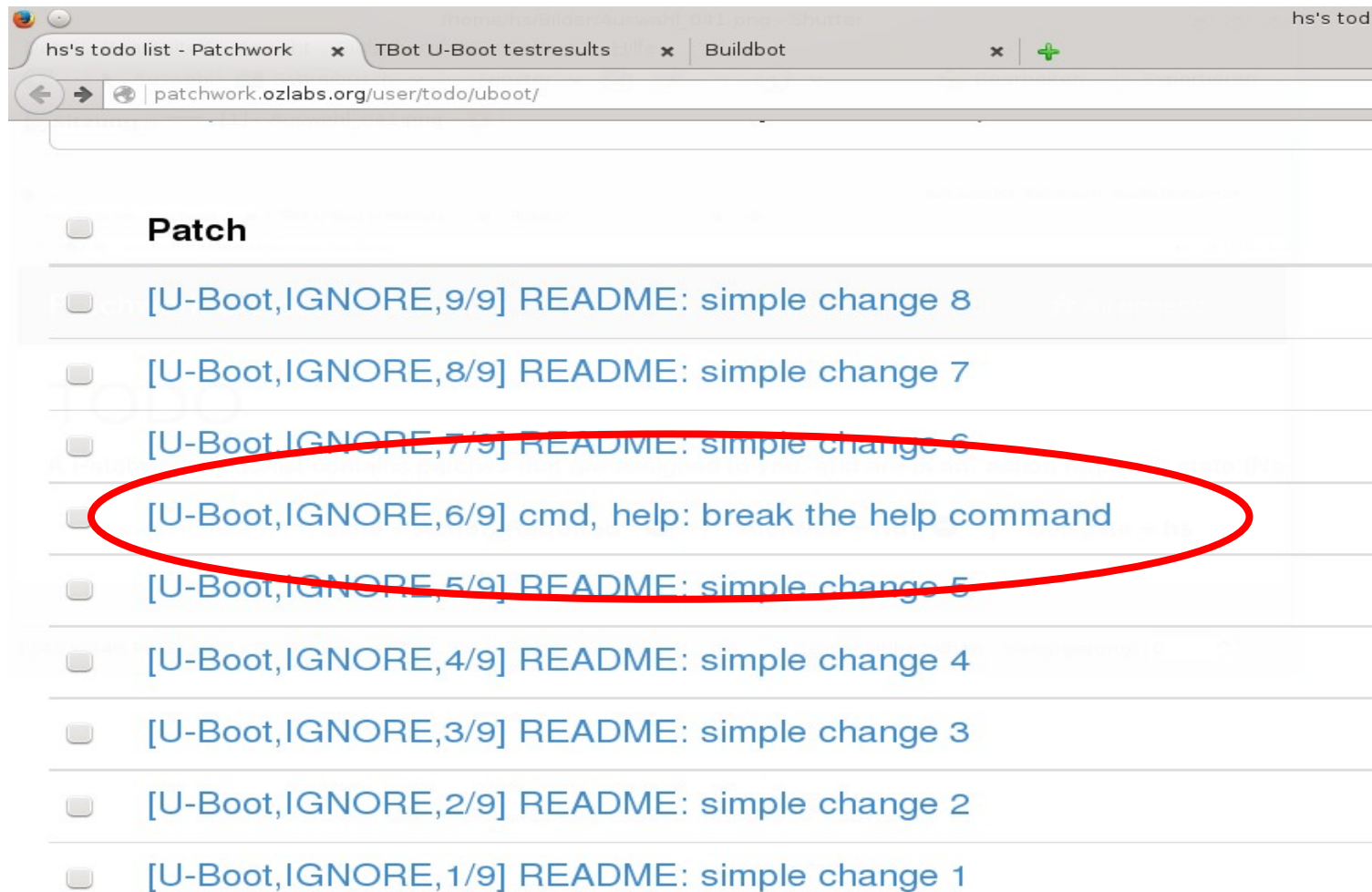
[src/tc/demo/tc\\_demo\\_part2.py](#)

- rufe TC: [tc\\_workfd\\_get\\_patchwork\\_number\\_list.py](#) auf hole alle patches einer Patchwork ToDo Liste
- Rufe TC: [tc\\_workfd\\_apply\\_patchwork\\_patches.py](#) auf checke den Patch mit checkpatch.pl und wende den Patch auf den aktuellen Source Code an.
- rufe TC: [tc\\_demo\\_compile\\_install\\_test.py](#) von Schritt 1 auf



# Schritt 2

Meine U-Boot Patchwork ToDo Liste zum Zeitpunkt des Testes



# Schritt 2 (cnt.)

Der Patch

<http://patchwork.ozlabs.org/patch/682156/>

Patch [hide](#) | [download patch](#) | [download mbox](#)

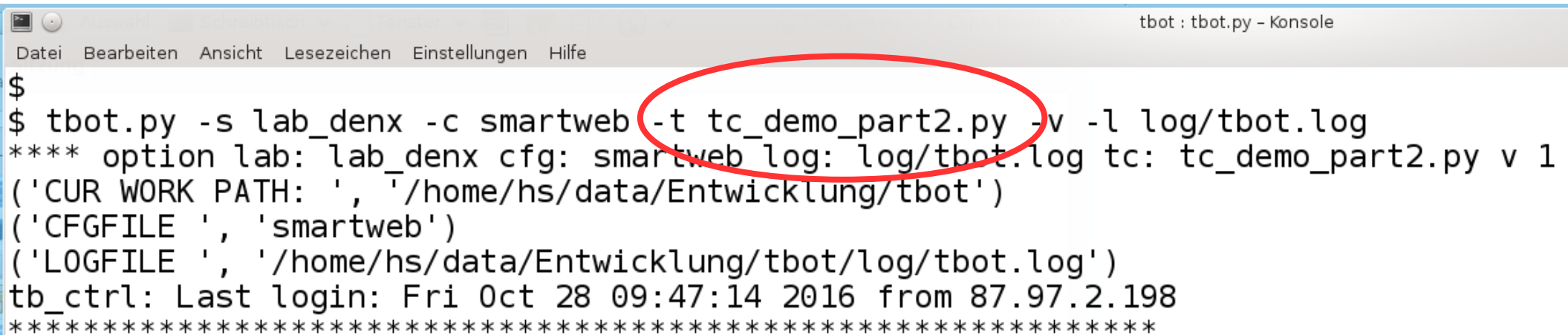
```
diff --git a/cmd/help.c b/cmd/help.c
index 701ae7e..d58bbdd 100644
--- a/cmd/help.c
+++ b/cmd/help.c
@@ -30,7 +30,7 @@ U_BOOT_CMD(

/* This does not use the U_BOOT_CMD macro as ? can't be used in symbol names */
ll_entry_declare(cmd_tbl_t, question_mark, cmd) = {
-    "?",    CONFIG_SYS_MAXARGS,    1,    do_help,
+    "??",   CONFIG_SYS_MAXARGS,    1,    do_help,
    "alias for 'help'",
#ifdef CONFIG_SYS_LONGHELP
    ""
```

... bringt den U-Boot help Kommand TC zum scheitern,  
Da der Patch bei der Ausgabe das eine „?“ in 2 „?“ ändert.

# Schritt 2 (cnt.)

Nun starten wir tbot ...



```
tbol : tbot.py - Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
$
$ tbot.py -s lab_denx -c smartweb -t tc_demo_part2.py -v -l log/tbot.log
*** option lab: lab_denx cfg: smartweb log: log/tbot.log tc: tc_demo_part2.py v 1
('CUR WORK PATH: ', '/home/hs/data/Entwicklung/tbot')
('CFGFILE ', 'smartweb')
('LOGFILE ', '/home/hs/data/Entwicklung/tbot/log/tbot.log')
tb_ctrl: Last login: Fri Oct 28 09:47:14 2016 from 87.97.2.198
*****
```

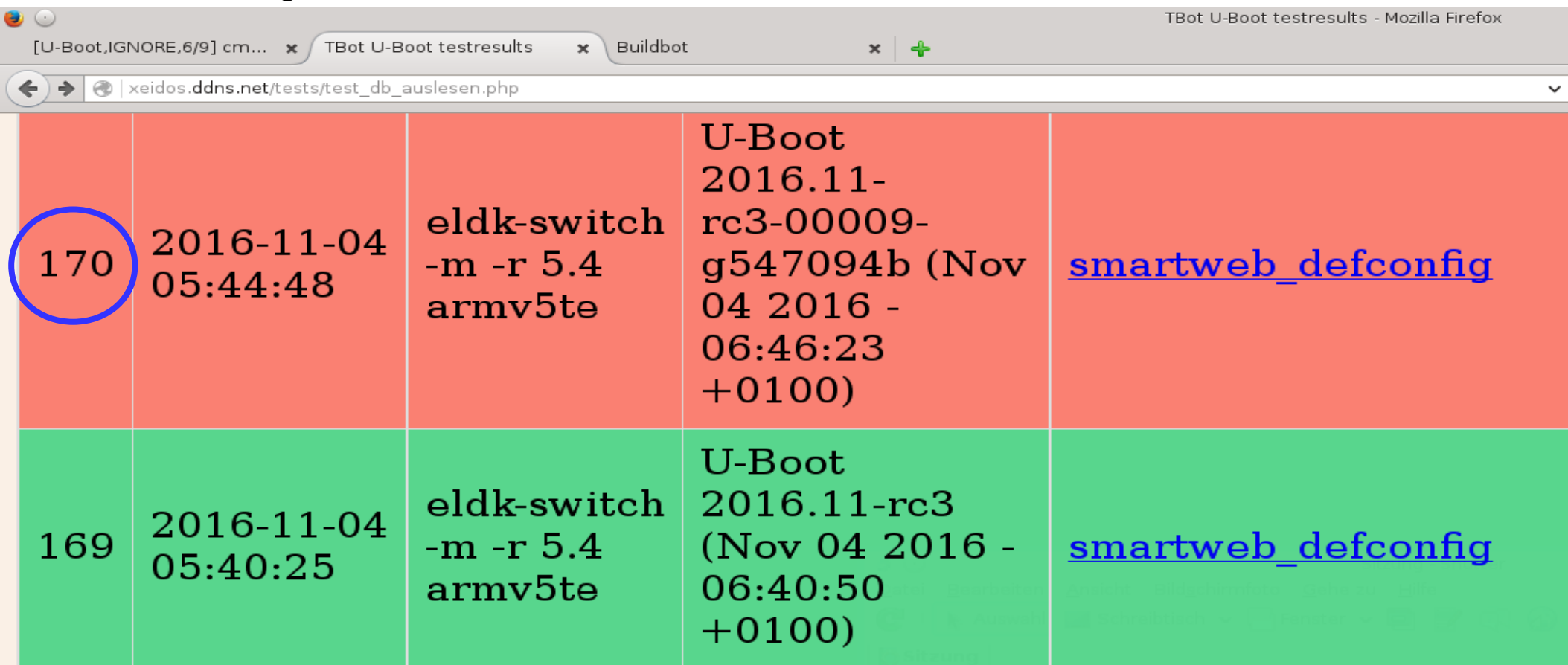
# Schritt 2: Ergebnis

```
setexpr - set environment variable as the result of eval expression
showvar - print local hushshell variables
sleep   - delay execution for some time
test    - minimal
tb_con: test like /bin/sh
tftpboot- boot image via network using TFTP protocol
true    - do nothing, successfully
usb     - USB sub-system
version - print monitor, compiler and linker version
U-Boot#
End of TB0T: failure
$ █
```

Wie erwartet endet tbot mit einem Fehler

# Schritt 2: Ergebnis

Schauen wir wieder auf unsere tbot Testergebnis Webseite.  
Und wir sollten nun eine neue Zeile mit einer neuen ID sehen, wobei der Hintergrund rot sein sollte.



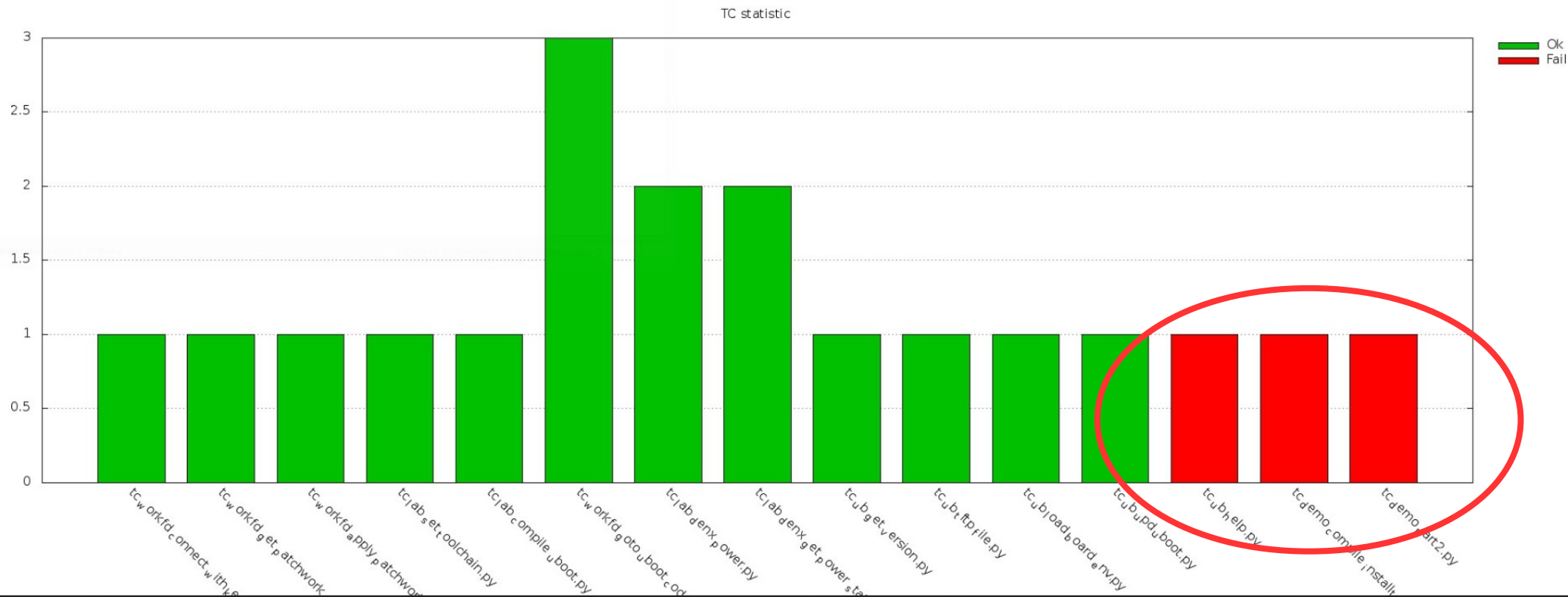
170	2016-11-04 05:44:48	eldk-switch -m -r 5.4 armv5te	U-Boot 2016.11- rc3-00009- g547094b (Nov 04 2016 - 06:46:23 +0100)	<a href="#">smartweb_defconfig</a>
169	2016-11-04 05:40:25	eldk-switch -m -r 5.4 armv5te	U-Boot 2016.11-rc3 (Nov 04 2016 - 06:40:50 +0100)	<a href="#">smartweb_defconfig</a>

[http://xeidos.ddns.net/tests/test\\_db\\_auslesen.php#170](http://xeidos.ddns.net/tests/test_db_auslesen.php#170)

# Schritt 2: Ergebnis

Nun schauen wir uns an, ob wir herausfinden können, warum der Test erfolglos war ...

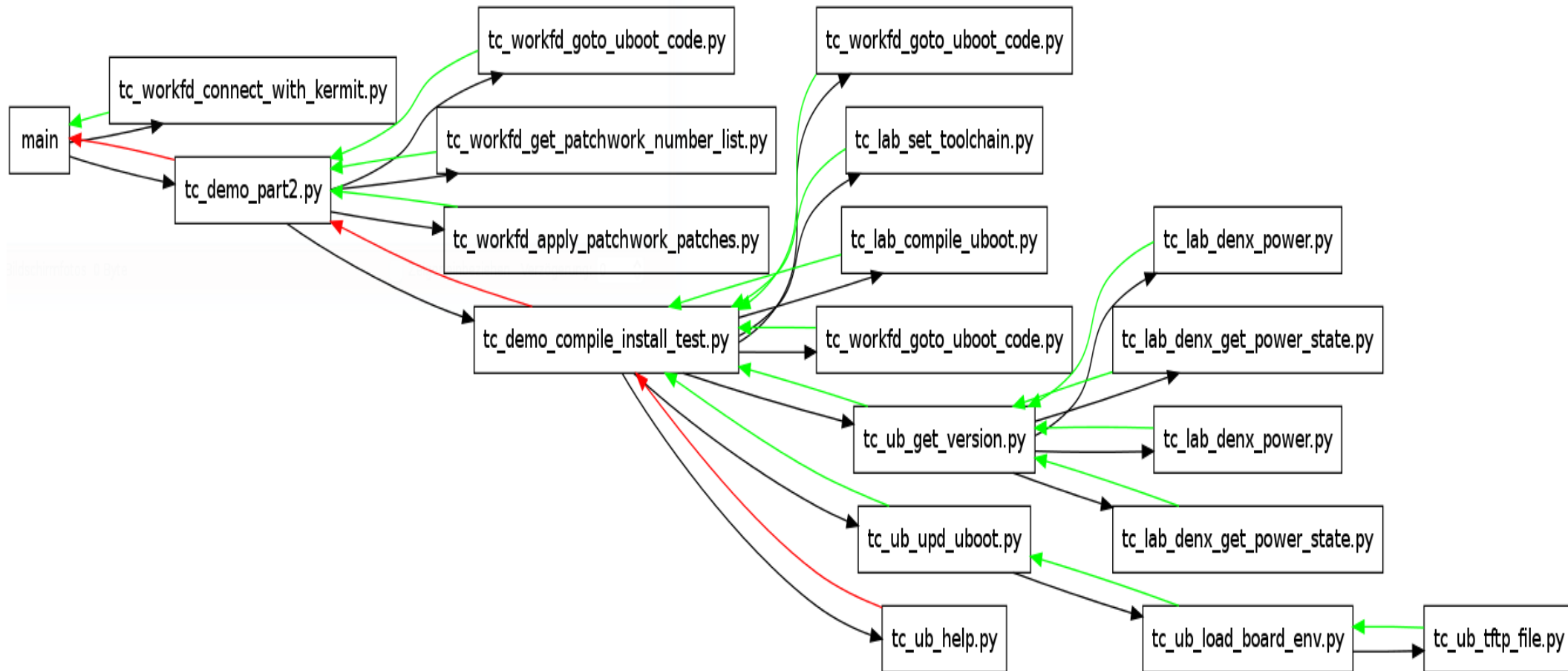
Zuerst schauen wir uns das statistic Bild an:



[http://xeidos.ddns.net/tbot/id\\_170/statistic.jpg](http://xeidos.ddns.net/tbot/id_170/statistic.jpg)

# Schritt 2: Ergebnis

Schauen wir noch auf das dot graph Bild



[http://xemos.ddns.net/tbot/id\\_170/graph.png](http://xemos.ddns.net/tbot/id_170/graph.png)

# Schritt 2: Ergebnis

Und noch in den „nice log“

```
[+] tc_ub_upd_uboot.py
[-] tc_ub_help.py
    [+] console
    Failed
    Failed
    Failed
OK
https://github.com/hsdenx/tbot/tree/master/src/tc/board/tc_dem
```

[http://xeidos.ddns.net/tbot/id\\_170/html\\_log.html](http://xeidos.ddns.net/tbot/id_170/html_log.html)

Alles deutet auf einen Fehler in [tc\\_ub\\_help.py](#) hin ...



# Schritt 2: Ergebnis

Können wir noch mehr herausfinden? ...

Ja, indem wir uns den Log zu dem U-Boot help TC anschauen

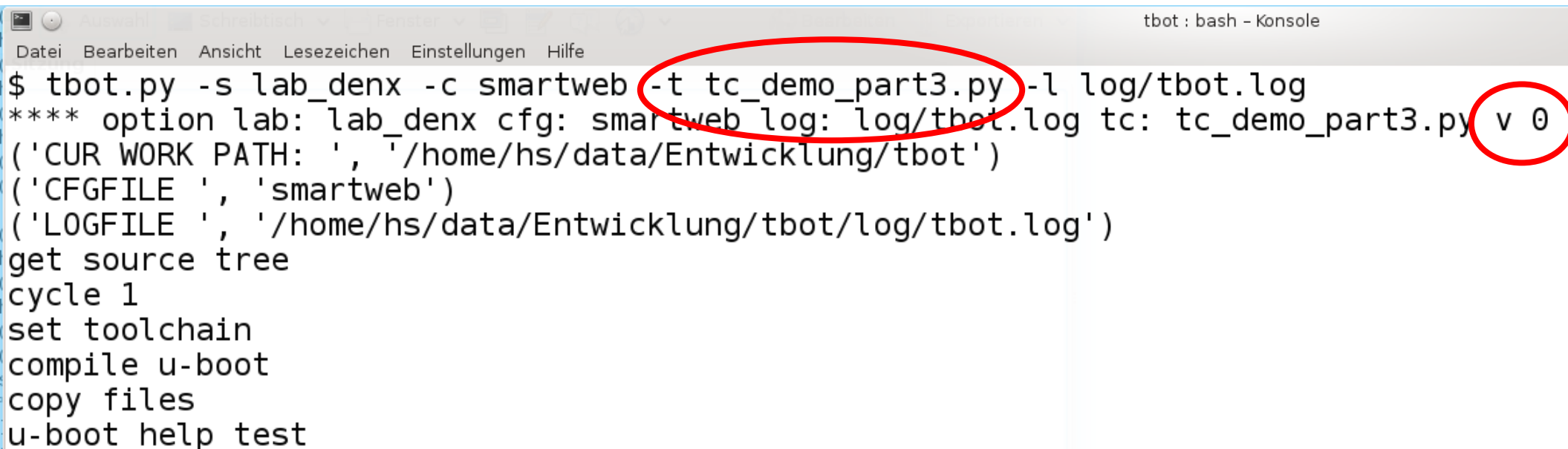
```
[ - ] tc_ub_help.py
[ - ] console
exception
U-Boot# help
?? - alias for 'help'
base - print or set address offset
bdinfo - print Board Info structure
boot - boot default, i.e., run 'bootcmd'
bootd - boot default, i.e., run 'bootcmd'
```

[http://xeidos.ddns.net/tbot/id\\_170/html\\_log.html](http://xeidos.ddns.net/tbot/id_170/html_log.html)

# Schritt 3

Nun starten wir noch den Testcase, mit dem wir automatisch herausfinden wollen, welcher der Patches den TC fuer das U-Boot help Kommando zum Fehlschlag bringt. Wir starten den TC ohne die verbose option, da der TC viele Ausgaben macht, sowie einige Zeit dauert ...

Testcase: [src/tc/demo/tc\\_demo\\_part3.py](#)



```
tbot : bash - Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
$ tbot.py -s lab_denx -c smartweb -t tc_demo_part3.py -l log/tbot.log
**** option lab: lab_denx cfg: smartweb log: log/tbot.log tc: tc_demo_part3.py v 0
('CUR WORK PATH: ', '/home/hs/data/Entwicklung/tbot')
('CFGFILE ', 'smartweb')
('LOGFILE ', '/home/hs/data/Entwicklung/tbot/log/tbot.log')
get source tree
cycle 1
set toolchain
compile u-boot
copy files
u-boot help test
```

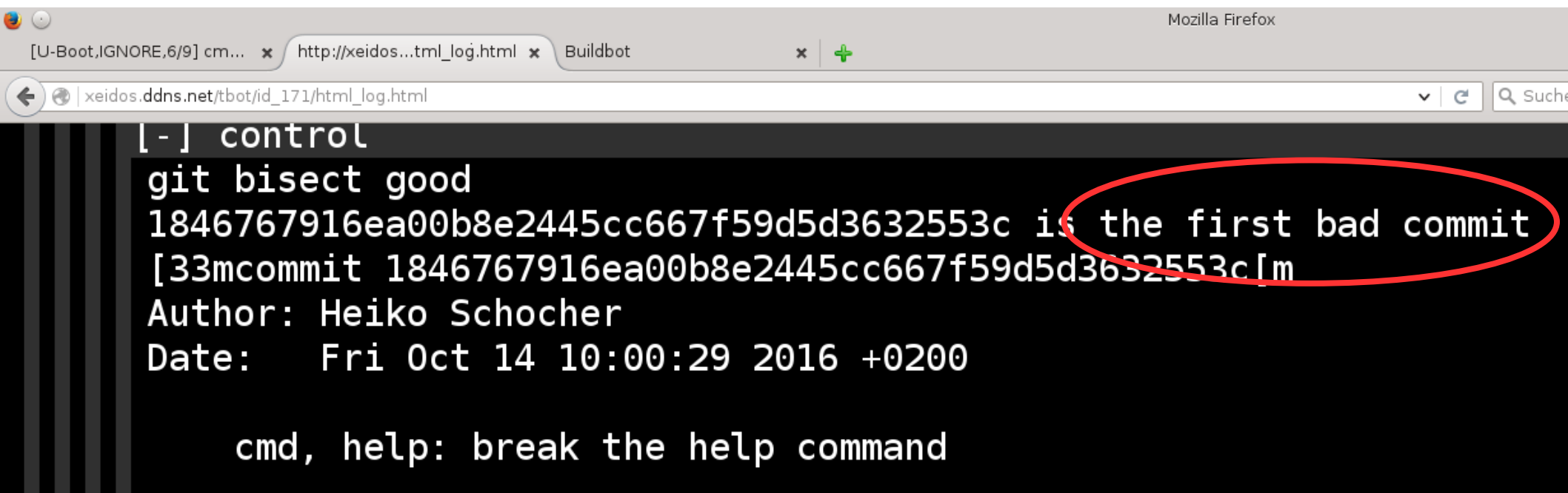
# Schritt 3: Ergebnis

```
cycle 2  
set toolchain  
compile u-boot  
copy files  
u-boot help test  
cycle 3  
set toolchain  
compile u-boot  
copy files  
u-boot help test  
End of TB0T: success
```

Prima, der TC endet mit Erfolg

# Step 3: Ergebnis

Nun schauen wir auf der tbot Ergebniswebseite in den „nice log“  
Wir sehen das Ergebnis des „git bisect“ Kommandos mit der  
Korrekten commit id, die den U-Boot help TC zum scheitern bringt



```
[U-Boot,IGNORE,6/9] cm... x http://xebidos...tml_log.html x Buildbot x +
xebidos.ddns.net/tbot/id_171/html_log.html
[-] control
git bisect good
1846767916ea00b8e2445cc667f59d5d3632553c is the first bad commit
[33mcommit 1846767916ea00b8e2445cc667f59d5d3632553c[m
Author: Heiko Schocher
Date: Fri Oct 14 10:00:29 2016 +0200

cmd, help: break the help command
```

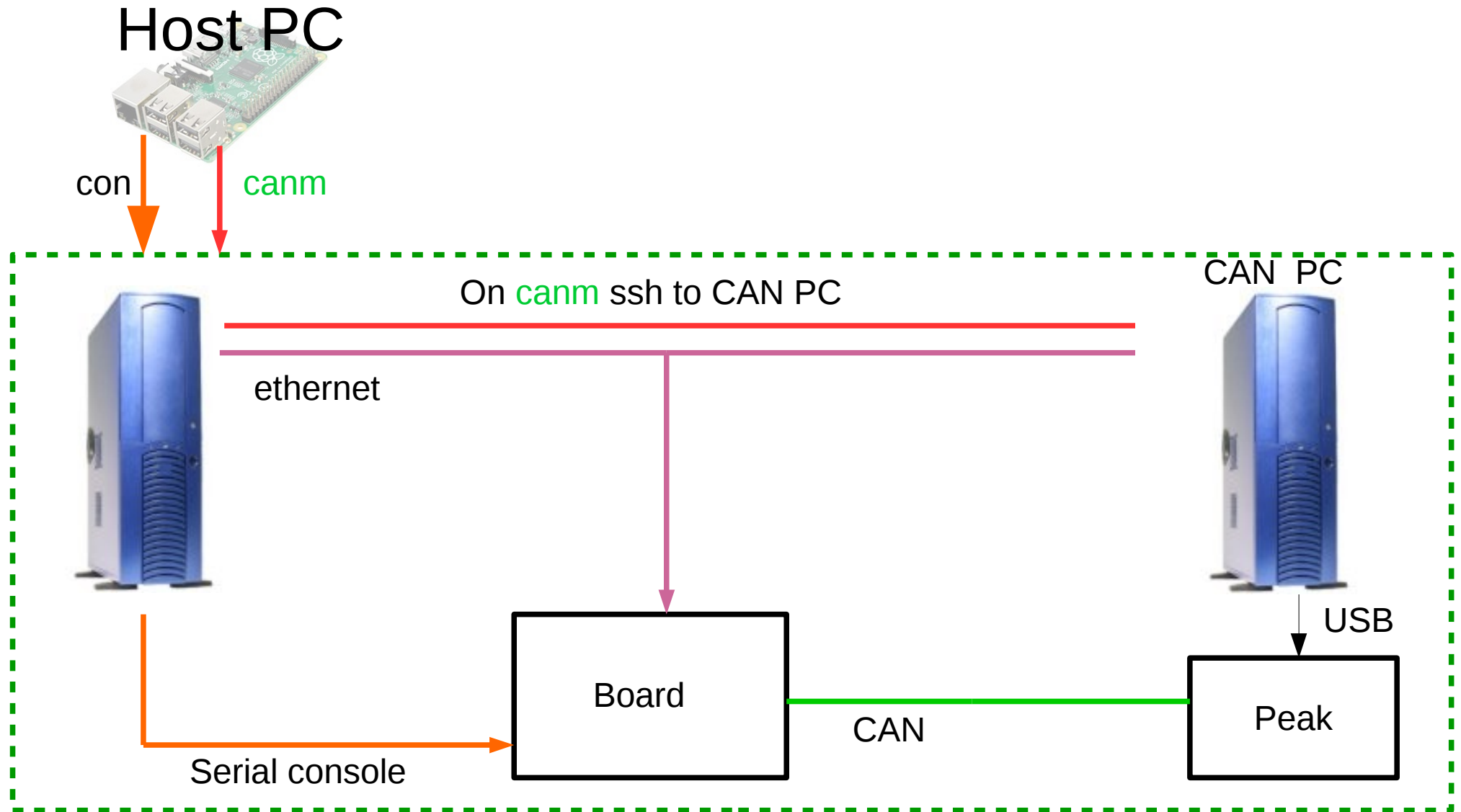
[http://xebidos.ddns.net/tbot/id\\_171/html\\_log.html](http://xebidos.ddns.net/tbot/id_171/html_log.html)

# Testcase: CAN bus

- Testen einer CAN bus Schnittstelle an einem Board
- Sie finden das komplette Video hier:

Video: <https://youtu.be/hl7gl4b9CG8>

# CAN bus TC demo standard VL Setup



# CAN bus TC demo (cnt.)

Starte TC:

[src/tc/demo/tc\\_demo\\_can\\_part1.py](#)

Was macht dieser TC:

- ruft TC: [tc\\_workfd\\_can.py](#) auf
  - öffnen einer neuen tbot Verbindung hier als **canm** bezeichnet
  - ssh über die **canm** zum CANPC
  - starten von candump über die **canm** Verbindung
  - senden von „0xdeadbeef“ über den canbus mittels des cansend Kommando über die console Verbindung.  
connection
  - prüfen, ob über die **canm** Verbinndung 0xdeadbeef empfangen wurde.

# CAN Bus start tbot

```
tbót : bash - Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
$
$
$
$ tbot.py -s lab_denx -c fipad -t tc_demo_can_part1.py -v -l log/tbot.log
**** option lab: lab_denx cfg: fipad log: log/tbot log tc: tc_demo_can_part1.py v 1
('CUR WORK PATH: ', '/home/hs/data/Entwicklung/tbot')
('CFGFILE ', 'fipad')
('LOGFILE ', '/home/hs/data/Entwicklung/tbot/log/tbot.log')
tb_ctrl: Last login: Fri Oct 28 10:26:56 2016 from 87.97.2.198
.....
```

Nach einiger Zeit, tbot endet mit Erfolg.

```
root@generic-armv7a-hf [16:12:00] ttbott>
tb_con: ./cansen
tb_con: d can0 123#DEADBEEF
tb_con: root@generic-armv7a-hf [16:12:00] ttbott>
tb_con: can0 123 [4] DE AD BE EF
End of TBOT: success
$ █
```



# Ergebnis des TC

```
[+] console
[+] control
[+] tc_lab_denx_connect_tc_board.py
[-] tc_demo_can_part1.py
[...]
```

```
[-] console
cd /home/hs/iproute2
root@generic-armv7a-hf [15:07:18] ttbot> ./ip/ip link set can0 type can bitrate 500000
RTNETLINK answers: Device or resource busy
root@generic-armv7a-hf [15:07:18] ttbot> ./ip/ip link set can0 up
root@generic-armv7a-hf [15:07:18] ttbot> cd /home/hs/can-utils
root@generic-armv7a-hf [15:07:19] ttbot> ./cansend can0 123#DEADBEEF
root@generic-armv7a-hf [15:07:19] ttbot>
```

```
[-] canm
ip link set can0 type can bitrate 500000
RTNETLINK answers: Device or resource busy
root@metis [15:07:17] ttbot> ip link set can0 up
root@metis [15:07:17] ttbot> cd /home/hs/can-utils
root@metis [15:07:18] ttbot> ./candump can0
can0 123 [4] DE AD BE EF
```

OK  
OK

# Links

- tbot source:  
<https://github.com/hsdenx/tbot>
- Running nightly builds with buildbot on a raspberry Pi at my home:  
<http://xeidos.ddns.net/buildbot/tgrid>
- Displaying test results  
[http://xeidos.ddns.net/tests/test\\_db\\_auslesen.php](http://xeidos.ddns.net/tests/test_db_auslesen.php)
- Demo für U-Boot test auf dem smartweb Board  
<https://youtu.be/zfjppj3DLsx4>
- CAN bus video  
<https://youtu.be/hl7gl4b9CG8>

# Fragen ?

Danke für Ihre Aufmerksamkeit!

Kontakt: Heiko Schocher <hs@denx.de>