Anleitung: Rebrandete Wlan-karte mit Linux nutzen

Beitrag von „Patricksworld“ vom 2. März 2017, 14:07

UPDATE: Ich habe die Schritte an einem 3 Cleaninstalls vorgenommen und es funktioniert jetzt fehlerfrei.

Der @Sascha_77 bietet hier an eure Wlankarten trotz Laptopwhitelist zum laufen zu bringen.

Für Windows und Linux war diese aber bisher dann für die Tonne. Da ich aber bekannter Weise Linux liebe und meinem Laptop im Dualboot benutze (OSX / Ubuntu) möchte ich allen anderen einen Weg zeigen die Karte dennoch unter Linux zu nutzen.

Da ich selber ein PC-Noob bin bitte ich um etwas Nachsicht, wenn hier etwas technisch nicht ganz korrekt erklärt wird. Schreibt einfach einen Hinweis und ich ändere die Fehler dann nachträglich.

Überblick Schritte:

- PCI-ID Überprüfen
- Kernel von kernel.org laden
- Kernel / Kernelmodule anpassen
- Kernel kompilieren
- (Grub2 anpassen - empfohlen, aber nicht nötig)
- Kernel installieren
- Postinstall / neuladen ath9k

Vorwort:

Da zum Kompilieren der Kernel gedownloadet werden muss, sowie einige Pakete vorab aus den Paketquellen geladen werden müssen, solltet ihr eine Internetverbindung über Lan oder

einen USB-Dongle herstellen.
Es werden ca 1,1 GB aus dem Internet geladen.

Spoiler anzeigen

Ich beziehe mich hier auf eine gerebrandete Atheros 9280 mit einer Intel N7260 PCI-ID und nutze Ubuntu.

Außerdem sollte jedem klar sein, das ihr nach einem Kernelupdate die Schritte erneut durchführen müsst!

Und um der VERWIRRUNG vorzubeugen.Ich benutze LINUX (UBUNTU) zum kompilieren. Es ist nur ein OSX-Theme. Also nicht das ihr auf die Idee kommt unter OSX zu kompilieren 

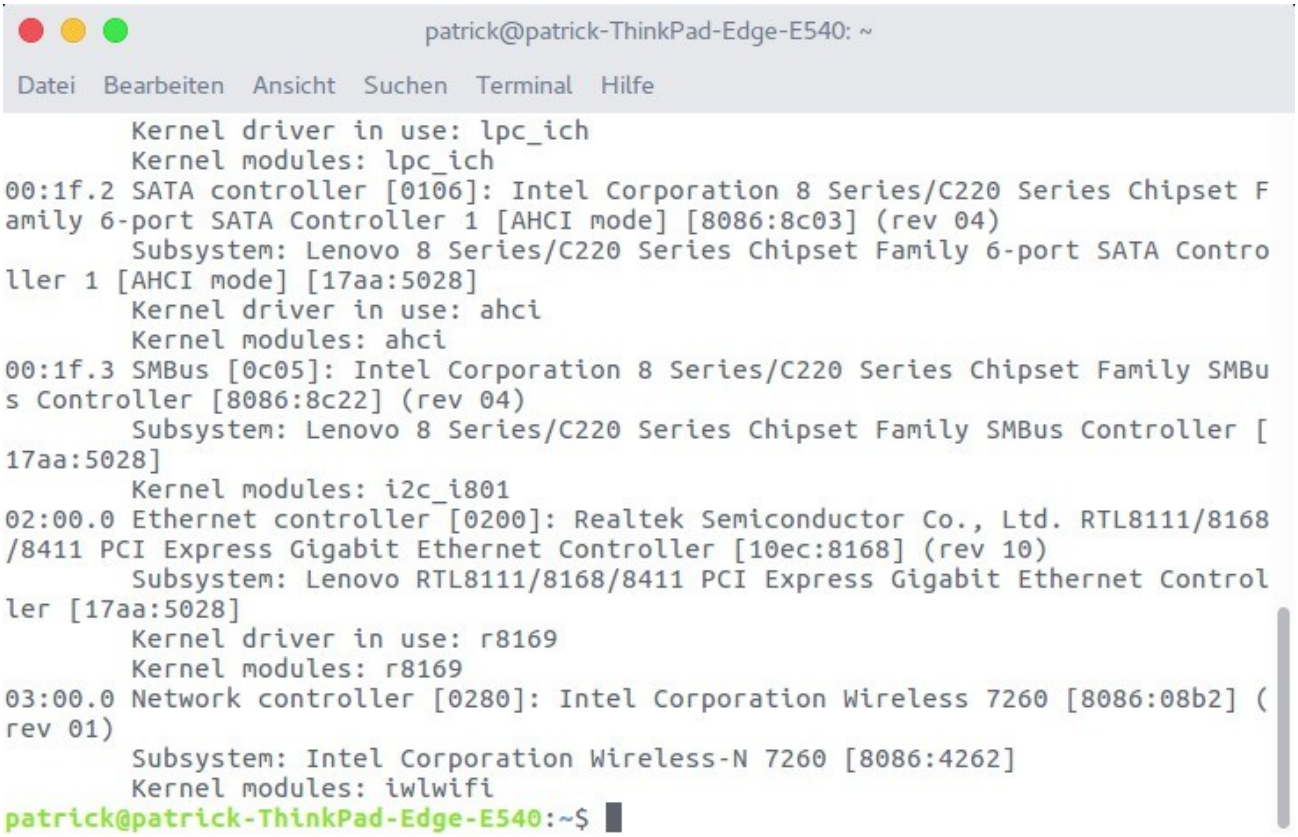
Schritt 1: PCI-ID Überprüfen

Ist die Karte getauscht könnt ihr zuerst nachsehen, was verbaut ist und welcher treiber läuft.

Das könnt ihr ganz einfach im Terminal überprüfen mit:

Code

```
1. lspci -nnk
```



Man erkennt dass das Kernelmodul (Treiber) iwlwifi geladen wird. Das wäre grundsätzlich ja auch korrekt für die entsprechende PCI-ID. Allerdings ist es unser Ziel jetzt den ath9k treiber zu laden, da dieser für die AR9280 zuständig ist. Dafür müssen wir den Kernel anpassen und neu kompilieren.

Schritt 2: Kernel downloaden

Öffnet ein Terminal und wechselt in euer Homeverzeichnis (eigentlich der standartordner)

Code

```
1. cd ~
```

legt euch einen neuen ordner an und wechselt in das Verzeichnis:

Code

```
1. mkdir kernelneu
2. cd kernelneu
```

den entsprechenden Kernel downloaden und entpacken (und umbenennen):

Code

```
1. wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.10.1.tar.xz
2. tar xf linux-4.10.1.tar.xz
3. mv linux-4.10.1 linux-4.10.1-custom
```

Anschließend in das Verzeichnis wechseln.

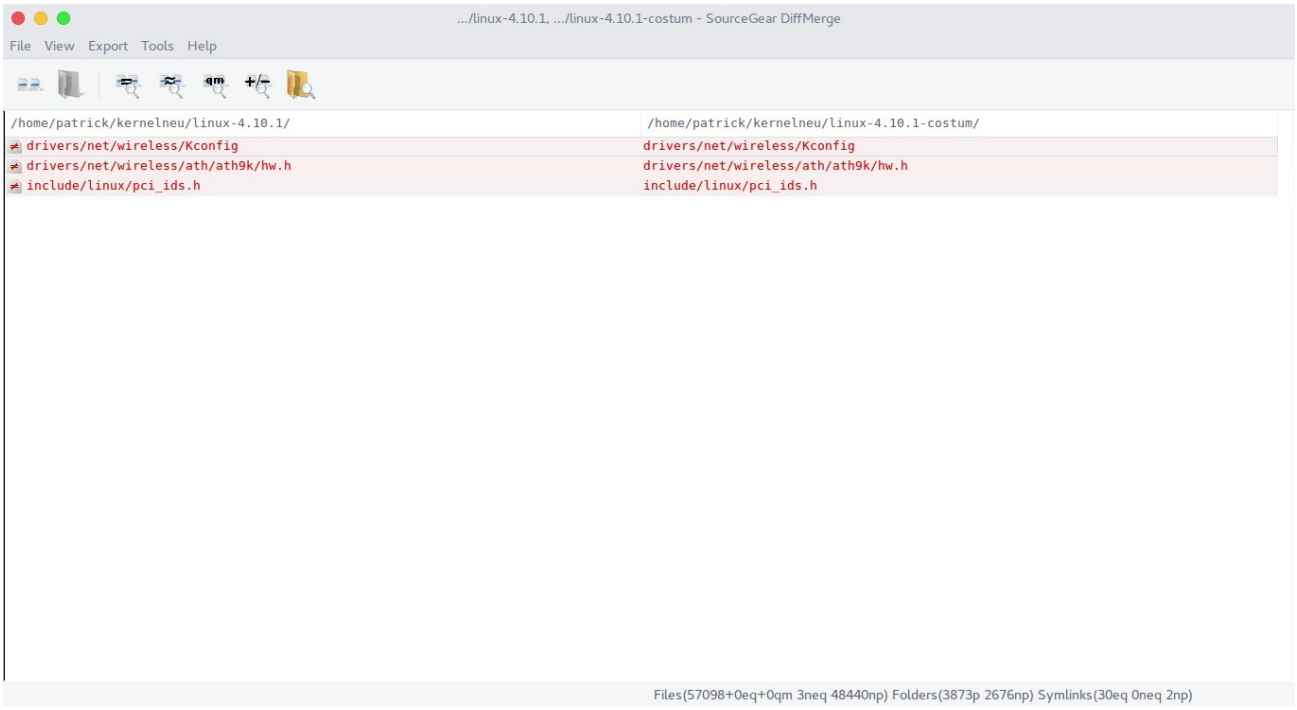
Code

```
1. cd linux-4.10.1-custom/
```

Schritt 3: Kernel / Kernelmodule anpassen

Ich habe lediglich 3 Dateien angepasst. Eine muss mit Sicheheit angepasst werden. Eine ist fakultativ und die 3 bin ich unsicher. Müsste man nocheinmal ohne testen.

Folgende 3 Datein habe ich angepasst:



Wir müssen mindestens im ath9k die pci-id anpassen. Unsicher bin ich mir mit den pci_ids.h Eintrag. Der stört aber auf jeden Fall nicht. Ich glaube aber es dürfte auch ohne gehen.

Also starten wir mit dem ath9k treiber.

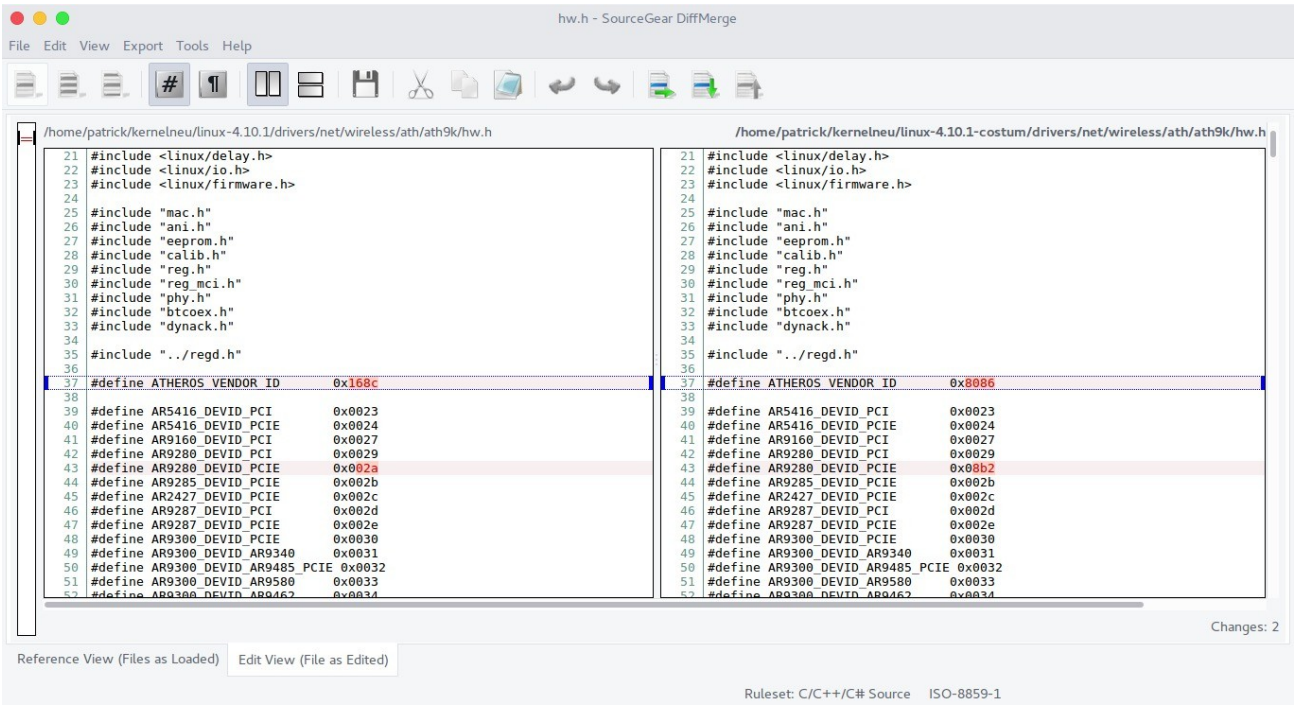
Die entscheidende Information befindet sich in der hw.h Datei. Also öffnen wir diese mit unserem Texteditors des Vertrauens (bei mir gedit)

Code

1. gedit drivers/net/wireless/ath/ath9k/hw.h

Dort tauscht ihr die entsprechenden PCI-ID's mit den vorher ermittelten und speichert das ganze ab.

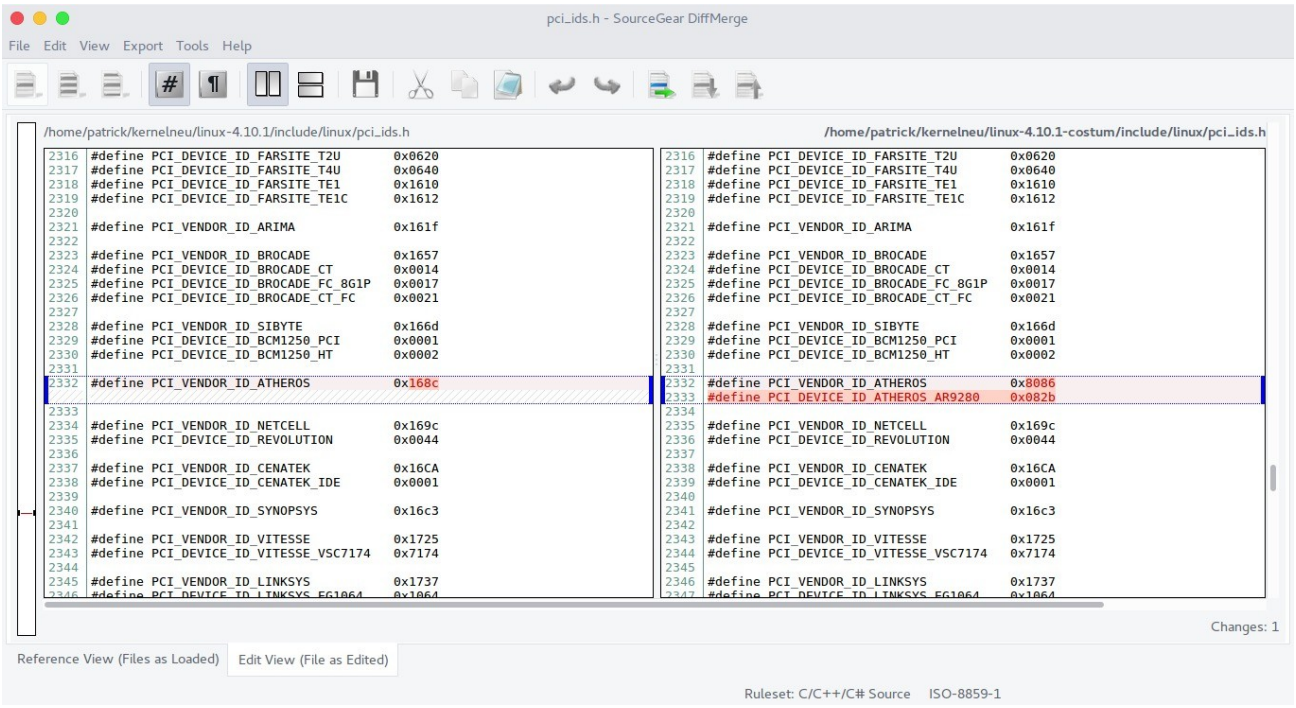
links original / rechts verändert



Als nächstes passen wir die pci_ids.h an:
(ich bin mir nicht sicher ob das zwangsläufig sein muss)

Code

1. gedit include/linux/pci_ids.h

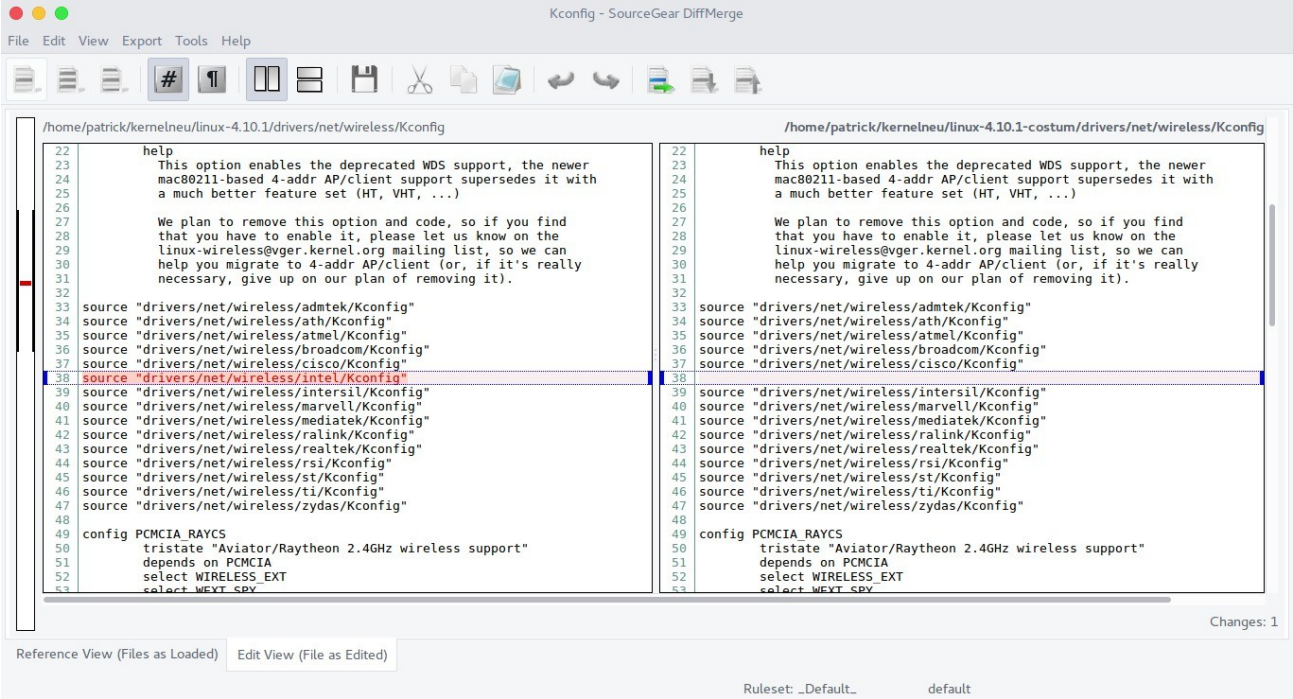


Und zu guter letzt passen wir noch die Kconfig an.

Das hat nur den Hintergrund das ich den ilwifi treiber garnicht mitkompilieren will. Somit kann das auch nicht fälschlicherweise geladen werden. Man könnte das Modul genauso gut nachträglich blockieren. Doch um mir den Ärger zu ersparen entferne ich einfach die entsprechende Zeile.

Code

1. gedit drivers/net/wireless/Kconfig



Und das waren auch schon die Anpassungen.

Der Rest ist Linux Standart Prozedur.

Schritt 4: Kernel Kompilieren

Generell benötigt ihr zum kompilieren folgende Pakete / Programme:

- linux-source
- build-essential
- kernel-package
- libssl-dev

Ihr könnt die einfach mit folgendem Befehl im Terminal laden und installieren:

Code

```
1. sudo apt-get install linux-source build-essential kernel-package libssl-dev
```

Zum Kompilieren müsst ihr folgende Befehle nacheinander ins Terminal reinschmeißen:

Code

```
1. cp /boot/config-`uname -r` .config
2. yes "" | make oldconfig
3. make -j8 bzImage modules
```

Schritt 5: Grub2 - anpassen (fakultativ)

So weit so gut. Vor dem Kernel installieren solltet ihr jedoch sicherstellen
das ihr zur not den "alten" Kernel laden könnt. Damit ihr auch ein Auswahlmenü bekommt um im Notfall den Kernel manuell auswählen zu können, benutze ich immer Grub-Customizer um wenigstens 10 sekunden zur Auswahl zu haben.

Installation im Terminal:

Fremdpaketquelle freischalten:

Code

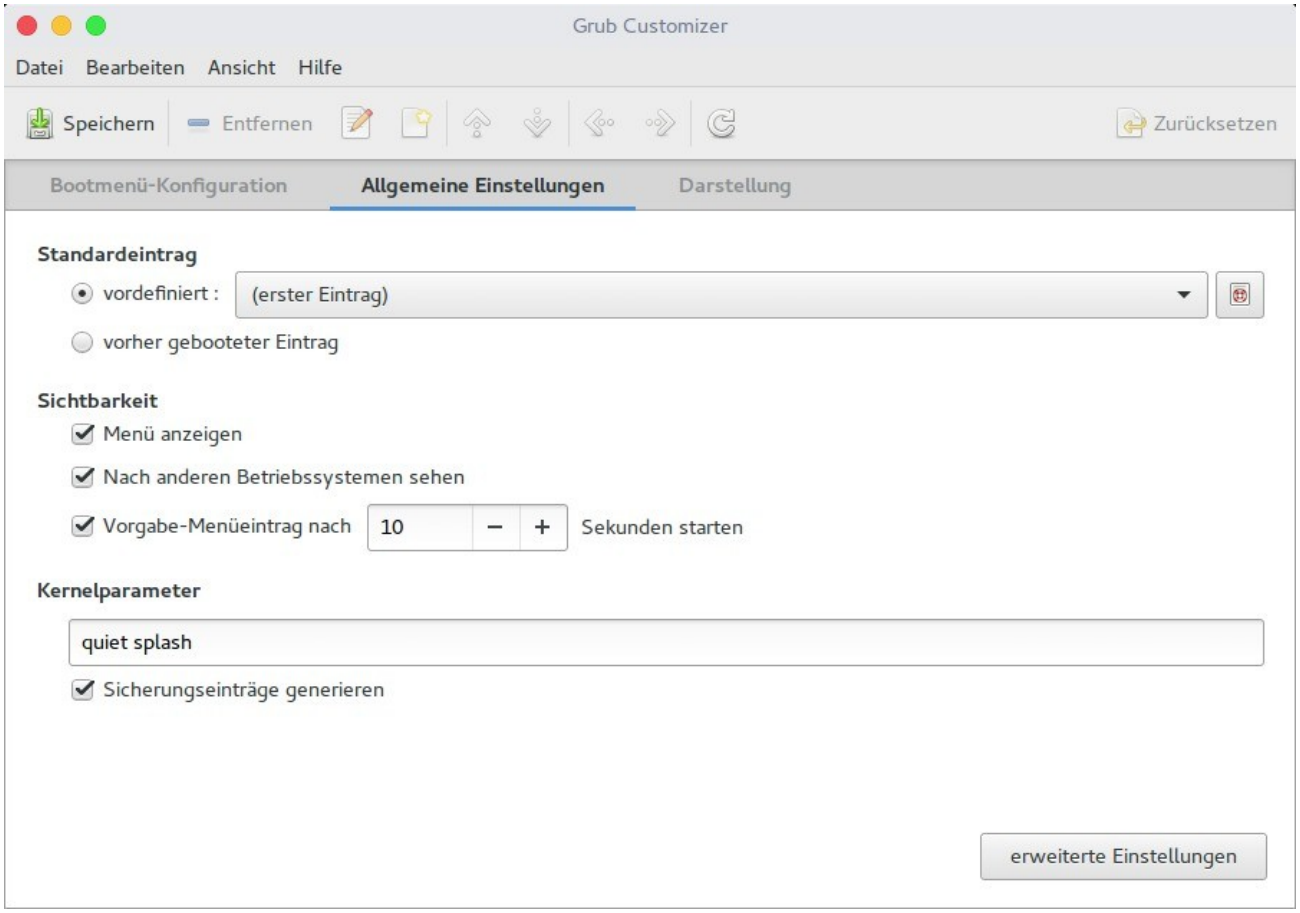
```
1. sudo add-apt-repository ppa:danielrichter2007/grub-customizer
```

Und installieren:

Code

```
1. sudo apt-get update
2. sudo apt-get install grub-customizer
```

Anschließend folgende Einstellungen übernehmen:



Das ist wie gesagt nur für Angsthasen

Schritt 6: Kernel installieren

fast geschaft:

Code

1. sudo make modules_install install

--> Neustart

Schritt 7: Postinstall / neuladen ath9k

ath9k konfigurieren/ Device anlegen:

Code

1. echo 'install ath9k /sbin/modprobe --ignore-install ath9k; /bin/echo "8086 08b2" > /sys/bus/pci/drivers/ath9k/new_id' | sudo tee /etc/modprobe.d/ath9k_options.conf

den alten ath9k entfernen und neu laden

Code

1. sudo modprobe -rfv ath9k
2. sudo modprobe -v ath9k

Dannach sollte es schon funktionieren.

Damit der ath9k bei systemstart automatisch geladen wird führen wir noch folgenden code aus:

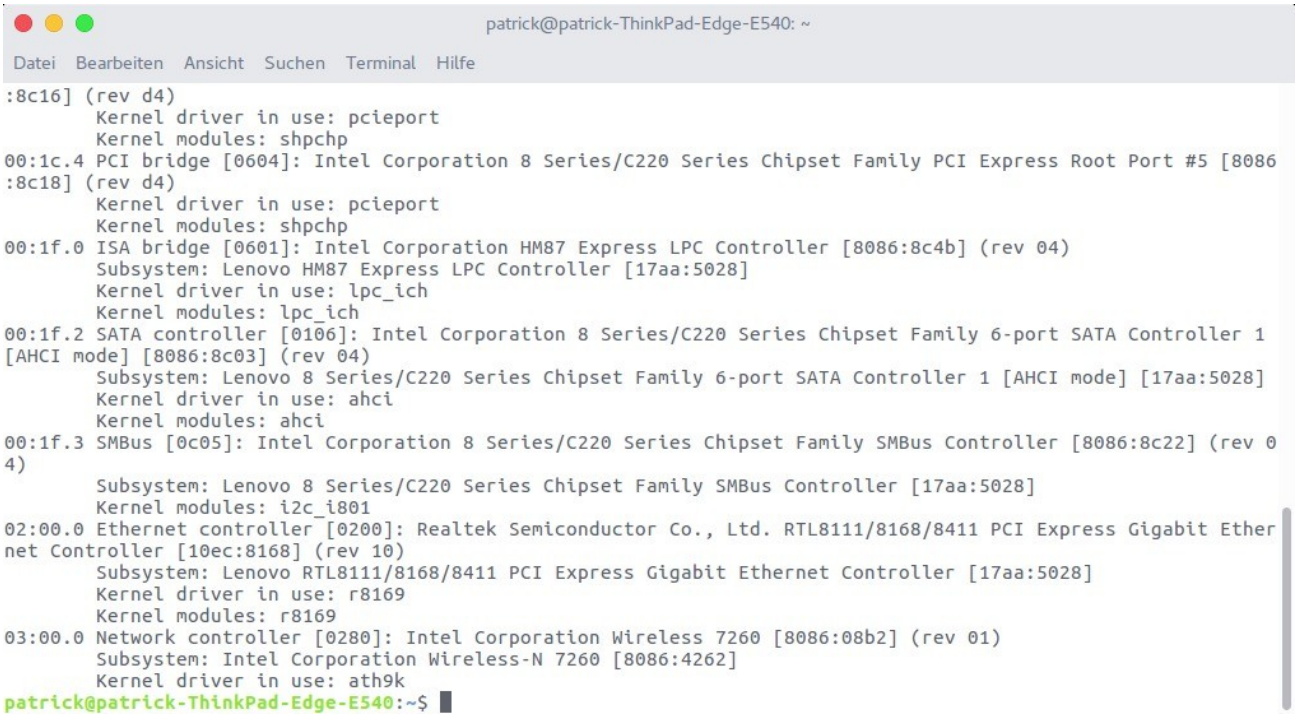
Code

1. echo ath9k | sudo tee -a /etc/modules

Überprüfen könnt ihr das mit einem erneuten:

Code

1. lspci -nnk



Ich freue mich auf Rückmeldungen.

MFG Patrick

Beitrag von „al6042“ vom 2. März 2017, 14:47

Alter Schwede...
Vielen Dank für die tolle Anleitung... Ich sehe mich schon Linux mit auf den T530 als Dualboot zu installieren, nur um in den Genuss der Möglichkeit zu kommen...

Tolle Arbeit und sehr gut erklärt...

Beitrag von „Patricksworld“ vom 2. März 2017, 14:51

Na dann leg mal los. Mich würde nämlich brennend interessieren ob es ohne UDEV-Regel funktioniert oder eben nicht. Bei mir ging es gerade ohne. Aber wie gesagt. Mein System ist leider nicht frisch.

Also hau mal in die Tasten und teste es.

MFG Patrick

Beitrag von „al6042“ vom 2. März 2017, 15:29

Bereite eben den Stick per unetbootin vor...

Beitrag von „Sascha_77“ vom 6. März 2017, 10:56

Wow danke. Super Anleitung. Da kann die Aktion "Wiederbeleben" ja starten.

EDIT:
Hab mich gestern mal unter Linux Mint daran begeben. Kernel kompilieren hat soweit geklappt. Allerdings hat er dann auf den letzten metern (als es darum ging den Kernel zu installieren) mit einer Fehlermeldung abgebrochen. Werde es heute mal mit Ubuntu versuchen.

Welches System nutzt du? @Patricksworld

Beitrag von „Patricksworld“ vom 6. März 2017, 19:28

Das Testsystem / Produktivsystem ist Ubuntu Gnome 16.04. Sollte aber eigentlich geauso mit Mint funktionieren. Wie war denn die Fehlermeldung? Der @al6042 hatte auch Mint im Einsatzt. Keine Ahnung was bei ihm rausgekommen ist.

MFG Patrick

Beitrag von „Sascha_77“ vom 6. März 2017, 19:58

Soooo. Nachdem ich jetzt alles mit Ubuntu 16.04 gemacht habe kann ich vermelden .... nicht nur das ich die Karte jetzt nutzen kann ... nein auch konnte ich mittels flash tool (wo es in den Sourcen die IDs ebenfalls umgebogen haben wollte) das original rom wieder zurückspielen. Die Karte lebt wieder. Jetzt kommt die 2. dran die ich hier rumliegen habe!

Beitrag von „al6042“ vom 6. März 2017, 20:01

Bei mir hat das kompilieren und installieren gut funktioniert, aber das Ergebnis war eher negativ...

Beitrag von „Patricksworld“ vom 6. März 2017, 20:03

Hast du die "postinstall" Schritte gemacht? Die hab ich erst nachträglich hinzugefügt.

Edit: Bei @Sascha_77 hat es unter Mint, warum auch immer, auch nicht funktioniert. Versuche es einmal mit Ubuntu 16.04 LTS.

mfg Patrick

Beitrag von „Sascha_77“ vom 6. März 2017, 20:07

Jop Mint scheint da etwas zickig zu sein.

Beitrag von „Patricksworld“ vom 6. März 2017, 20:11

Es erscheint eigentlich wenig logisch, da der Kernel ja unabhängig von Mint oder Ubuntu ist, aber offensichtlich hängt da doch irgendwo der Hase im Pfeffer. Wenn ich einmal Zeit, Lust und gut Laune habe, versuche ich es vielleicht auch einmal damit.

Beitrag von „Sascha_77“ vom 6. März 2017, 20:16

Die zweite Karte lebt nun auch wieder. Obwohl hier sogar die SUB IDs verbogen waren (bei der ersten nur die Haupt-IDs) konnte ich das Rom Problemlos zurückschreiben. Die SUB-Id scheint für Linux somit scheinbar generell keine Rolle zu spielen.

Beitrag von „Patricksworld“ vom 6. März 2017, 20:18

Nope. Das ist mir auch schon aufgefallen. Denn die wurden ja schließlich auch nicht verändert.

Wer braucht die schon ? 

Beitrag von „Sascha_77“ vom 28. März 2017, 22:34

Ich habe gerade herausgefunden, dass man nicht den ganzen Kernel kompilieren muss sondern man kann auch ein einzelnes Modul compilen. Im Fall der ath9k Treiber sähe das so aus:

Code

1. sudo make modules SUBDIRS=drivers/net/wireless/ath/ath9k
2. sudo make modules_install SUBDIRS=drivers/net/wireless/ath/ath9k

Das wars dann auch schon und spart ne Menge Zeit. Nichtmal ein Reboot ist notwenig. Die Karte wird dann direkt erkannt.

Beitrag von „grt“ vom 16. August 2017, 13:34

@Sascha_77 und/oder @Patricksworld grad hab ich erfogreich eine AR9285 rebrandet und ins

T430s verfrachtet. der kasten hat ein dualboot (osx/ubuntu) - müsste der kernel bzw. das modul bei jedem update neugemacht werden? oder gibts da irgendeinen automatismus, mit dem man das umgehen könnte? - son "gebastel" wär nämlich eindeutig userinkompatibel - (ist nicht meiner... )
ansonsten kriegt ubuntu einen kompatiblen usbwlanstick mitgeliefert....

Beitrag von „Sascha_77“ vom 16. August 2017, 13:57

Mit jedem Kernelupdate ist die Änderung wieder weg.

Beitrag von „grt“ vom 16. August 2017, 14:00

(grrr.. also usbwlanstick mitliefern.)
danke! 

Beitrag von „Sascha_77“ vom 16. August 2017, 14:06

grrr(t) ... leider. 

Beitrag von „grt“ vom 16. August 2017, 14:13

 selber 

Beitrag von „Sascha_77“ vom 16. August 2017, 15:50

Übrigens hatte ich die Tage ne Karte die ich verflasht hatte (statt 8086 dann 0686). Ich habe das Teil zum verrecken nicht ans laufen bekommen. Keine AHnung wieso. Ich habe ja schon 2 Karten recovern können. Warum es mit der nicht wollte ist ein Rätsel. Vllt. habe ich noch irgendwas anderes an Werten in den ersten 512 Byte des Chips erwischt.

Beitrag von „grt“ vom 16. August 2017, 17:18

uuups... ist mir beim ersten versuch auch passiert, zahlendreher gebaut - glücklicherweise bei den sub-ID's
liess sich reparieren (kann man auch karten zurückflashen, die schon eine andere VendorID haben? nebenbei gefragt) .

und sag mal, der kartoffelsalat_reloaded hatte in sein t430 eine atheros eingebaut, die offensichtlich gelistet war - ich hab die entsprechende subID auf eine atheros mit gleicher deviceID geflasht, aber das T430s war überhaupt nicht amused - kann es sein, dass T430 und T430s unterschiedliche listen haben?
es wär nämlich sowas von super gewesen, einfach nur die subID's ändern, und die karte wär auch im linux gerannt - nun ist es eine "intel"

Beitrag von „Sascha_77“ vom 16. August 2017, 17:51

Es reicht nie nur die SubIDS oder nur die HauptIDs zu ändern. Es muss immer beides geändert werden sonst wirds nix.

Das ist gut möglich das die untersch. Listen haben. Selbst in der selben Baureihe (T420) z.b. sind die unterschiedlich. So einen Fall hatte ich nämlich mal bei @Raoul Duke. Ich hatte damals ein T420 und er auch. Wollte eine WLan Karte haben. Also hab ich mein T420 hergenommen und die IDs davon geflasht. Beim Raoul im Rechner allerdings große Überraschung ... wollte er nicht schlucken. Und tats. hatte er eine andere orig. Karte als ich im Gerät.

Daher kann man (zum. bei Lenovo) nicht nur nach den Modellbezeichnungen gehen sondern muss immer individuell auslesen vorher.

Beitrag von „grt“ vom 16. August 2017, 18:04

> Zitat von Sascha_77
>
> Es reicht nie nur die SubIDS oder nur die HauptIDs zu ändern. Es muss immer beides geändert werden sonst wirds nix.

in dem fall dachte ich mir: in kartoffelsalats läppi soll eine AR9285 168c:002b:17aa:30a1 rennen - hier lag eine mit 168c:002b:144f:7167 - ergo in der firmwaredatei alle 4F14 6771 zu AA17 A130 ändern, zurückspielen, ab ins T430s und freuen... das linux auf dem flashrechner hat die umgestrickte karte erkannt, wollte sofort ins netz, terminal befragt: .... 17aa:30a1 - aber dann doch nicht in der liste :(

nunjut...

Beitrag von „Patricksworld“ vom 16. August 2017, 23:32

> Zitat von grt
>
> oder gibts da irgendeinen automatismus, mit dem man das umgehen könnte?

mit einem kleinen sh script sollte das eigentlich auch kein großes ding sein. Wenn du magst kann ich dir gerne mal versuchen eins zu basteln. Aber das geht dann halt nur spezifisch für den Lappi. Aber an sich klingt das nicht nach der großen Hürde.

Beitrag von „grt“ vom 17. August 2017, 10:32

@Patricksworld gerne! der läppi ist zwar gestern umgezogen, und sein herrchen ist grundsätzlich mit dem usbwlan einverstanden, aber bei nur 3 usb-anschlüssen wärs schon schön, den einen auch noch frei zu kriegen. und interessant wärs sowieso. aber meine shellscriptingkenntnisse sind ein wenig begrenzt...
ich mach mich mal auf die jagd nach einer umstrickbaren karte zum testen in einem von meinen läppis.

Beitrag von „Andy51105“ vom 21. März 2018, 09:22

@Patricksworld Tolle Anleitung hast du da erstellt. 👍

Ich habe das ganze mit einem X230 (I5 3320) und einer Wlan-Karte von Sascha_77 gemacht. Leider hat es bei mir nicht ganz funktioniert.

Installiert habe ich Ubuntu 16.04 und als Kernel wird mir 4.13.0-36/37 angezeigt. Ich habe daraufhin den Kernel 4.13.1 runtergeladen und die Aktion damit gestartet. (Ist für das X230 eine richtige Lebensaufgabe)

Soweit läuft auch alles durch, aber es wird am Ende beim Überpüfen mit "lspci -nnk" immernoch "Kernel driver in use: iwlwifi" und "Kernel modules: iwlwifi" angezeigt.

Ich habe das ganze dann auch mal mit dem kernel 4.10.1 probiert, mit dem gleichen Ergebniss.

Wo liegt der Fehler bei mir? (Ich weiß, sitzt vor dem PC 👍)

EDIT:

Nach der 4. Wiederholung hat es nun endlich funktioniert und läuft prima.

Nochmal danke für die tolle Anleitung. 

Beitrag von „griven“ vom 26. März 2018, 23:30

Nice, ich liebe es wenn Anleitungen funktionieren so soll es sein

---

Beitrag von „Andy51105“ vom 27. März 2018, 15:23

Ich hätte da noch 1 bis 2 bis 3 Fragen . 

Wenn ich die BackUp-Funktion (ähnlich wie TimeMachine) in Ubuntu nutze, kann ich dann alles wieder Herstellen und es läuft sofort wieder, oder muss ich diese Prozedur jedesmal neu machen?

Hab nämlich eine neue SSD eingebaut und sitze gerade beim kompilieren. Macht mit dem X230 keinen Spaß

Beitrag von „siegertyp“ vom 30. März 2019, 08:56

Morgen

Ich muss den Thread nochmal ausgraben. Erstmal, vielen Dank für die Arbeit, die du dir hiermit gemacht hast!

Allerdings funktioniert die Anleitung bei mir nicht wie geschildert. Egal wie ich es versuche, das Kompilieren des Treibers (oder auch des ganzen Kernels) schlägt (immer!) mit Error 2 fehl. Ich weiß nicht was ich noch probieren könnte, das T430 hat die ganze Nacht gerödelt und es ist gar nichts dabie raus gekommen, das ist deprimierend. Vielleicht kann mir hier ja jemand helfen, unter Umständen funktioniert der hier angegebene Kernel auch gar nicht mehr unter 18.04...

Beitrag von „Patricksworld“ vom 31. März 2019, 11:21

Naja. Ich würde immer den aktuellesten Kernel benutzen. Habe das ganze auch unter Ubuntu 18.04 getestet. Läuft ganz genauso ab. Aber wenn du hilfe Brauchst und ein Lankabel zur hand hast, dann können wir das gerne zusammen via Teamviewer machen.

MFG Patrick

Beitrag von „siegertyp“ vom 31. März 2019, 19:19

Vielen Dank für deine Hilfe bzw. den entscheidenden Denkanstoß. Nach Stunden langem Kompilieren läuft der aktuellste Kernel jetzt einwandfrei. Das Problem war das Verwenden eines uralten Kernels (dem in der Anleitung angegebenen ).