Raspberry PI – Cross Compiling unter Mac OS X mit Eclipse

Post an 26. Januar 2013 von Knut Tagged: C/C++, Linux, Raspberry PI, Tutorial

Primär beschreibe ich hier das Cross Compiling für den Raspberry PI unter Mac OS X mit der Entwicklungsumgebung Eclipse. Die Anleitung sollte auch unter Linux und Windows funktionieren, jedoch unterscheidet sich die Installation der ARM-Toolchain.

Installation der ARM-Toolchain

Bei dem Versuch die Sources selbst zu kompilieren, bin ich irgendwann auf die Seite Sourcery G++ Lite for ARM GNU Linux (2009q3 67) for Mac OS X gestoßen. Sie beinhaltet neben einer exakten Anleitung zur Kompilation der Toolchain auch alle Quellen und Patches. Was aber noch besser ist, die Toolchain können dort bereits kompiliert geladen werden. Für die ersten kleineren Programme sollten diese also ausreichen, wie weit man damit kommt habe ich noch nicht ausgetestet. Der Download-Link lautet:

- https://github.com/downloads/UnhandledException/ARMx/ARMx-2009q3-67.tar.bz2

Das Archiv downloaden, entpacken und in das Verzeichnis "/opt" verschieben:

```
1 :Downloads $ tar zxvf ARMx-2009q3-67.tar.bz2
2 :Downloads $ sudo mv ARMx-2009q3-67 /opt/arm-linux-tools
```

(Es kann auch innerhalb des Home-Verzeichnisses verbleiben, dies ist nur meine persönliche Präferenz.)

Damit die Toolchain auch aus dem Terminal, ohne voranstellen von "/opt/arm-linux-tools", ausgeführt werden können, muss der Pfad noch der Umgebungsvariable mitgeteilt werden. Da `export PATH=/opt/arm-linux-tools:$PATH` nur temporär gültig ist und so nach jedem Aufruf des Terminals neu eingegebne werden müsste, ist es einfacherer die Pfadangabe in die Konfigurationsdatei einzutragen:

```
1 :~ $ sudo nano /etc/paths
```

Am Ende der Datei die Pfad inklusive "/bin" hinzufügen:

```
1 /usr/bin
2 /bin
3 /usr/sbin
4 /sbin
5 /usr/local/bin
6 /opt/arm-linux-tools/bin
```

Eclipse installieren und einrichten

Nun muss, sofern noch nicht vorhanden, Eclipse IDE geladen und installiert werden. Die aktuellen stabile Version sind unter dem folgenden Link verfügbar: http://www.eclipse.org/downloads/

Ich habe Eclipse IDE for C/C++ Developers Juno in der 64Bit Version verwendet. Welche Version ist letztlich egal, es muss jedoch das Eclipse CDT (C/C++ Development Tooling) Plugin verfügbar sein.

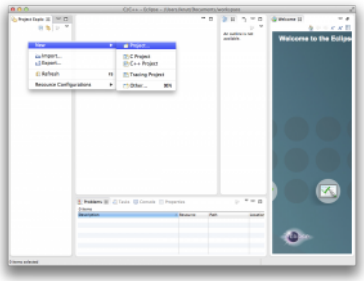
Nach dem der Download-Prozess abgeschlossen ist, das Archiv mit einem Doppelklick entpacken, das Verzeichnis “Eclipse” in den Programmordner legen und mit Doppelklick auf **Eclipse** öffnen.

Ach wenn Eclipse frisch installiert ist, sollte nun zuerst überprüft werden ob neue Updates zur Verfügung stehen. Hierfür das Menü **Help** auswählen und auf den Eintrag **Check for Updates** Klicken.

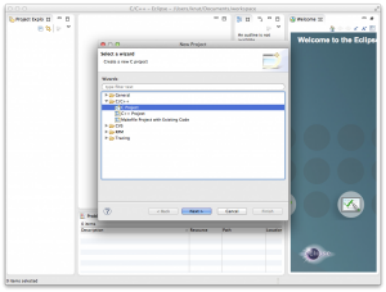
Sollte ein neues Update verfügbar sein, diese mit **Next** installieren.

Das erste Projekt

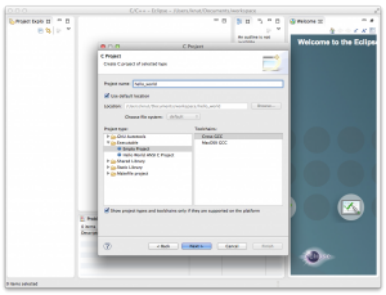
Ob alles funktioniert, überprüft man am einfachsten mit einem kleinen “Hello World” Programm. Dazu das Menü **File → New → Project** wähle.



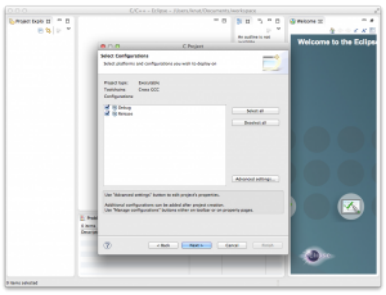
Als Projekt Art **C Project** auswählen.



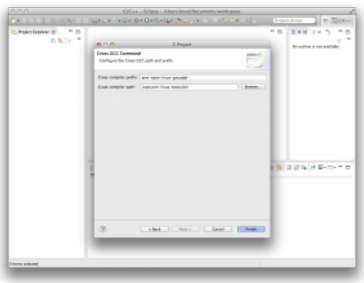
Dem Projekt einen Namen geben und als Type **Executeable → Empty Project → Cross GCC** wählen.



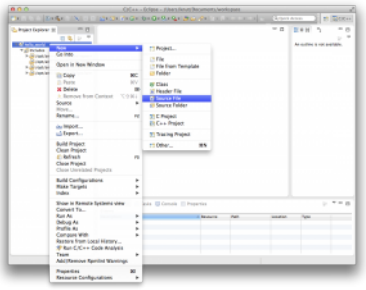
Eclipse bietet immer zwei Konfigurationen, an ein für den Debug und eine für das abschließende Release. Es kann einfach so übernommen werden.



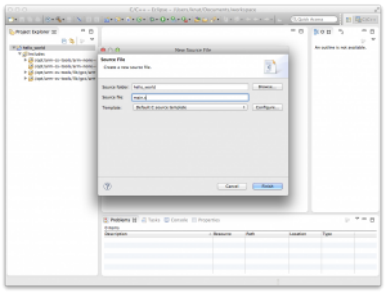
Auch wenn der Compiler im Pfad angegeben wurde, muss er noch Eclipse mitgeteilt werde, da Eclipse nicht weiss welchen Cross Compiler verwenden werden soll. Es muss allerdings nur der Prefix **arm-none-linux-gnueabi-**, so wie der Pfade zu diesem (“/opt/arm-linux-tools/bin”) angegeben werden. Je nach Projekttype wird gcc oder g++ ergänzt. Anhand des Pfades wird noch mitgeteilt, wo die Include und Lib Dateien liegen.



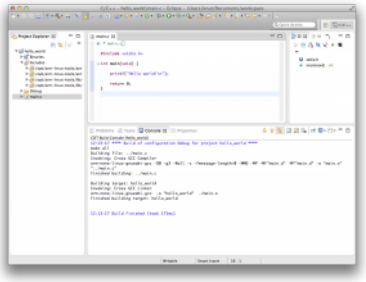
Ist das Projekt erstellt, muss diesem noch eine Source Datei zugeordnet werden.



Zum Beispiel eine Datei mit dem Namen “main.c”.



In die “main.c” kann nun der “Hello world”-Source eingefügt werden.



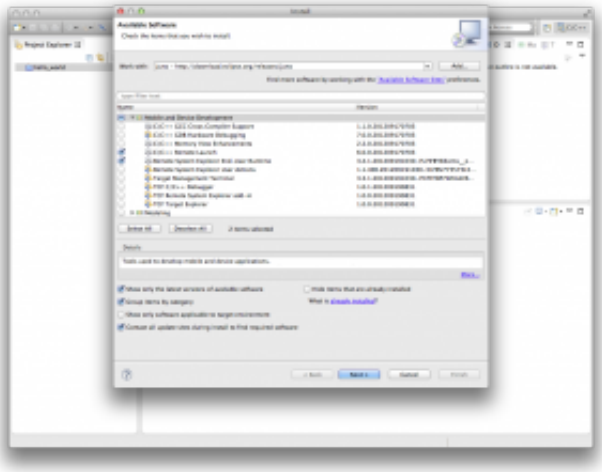
Über das Menü **Project → Build Project** das Programm kompilieren. Es befindet sich danach im Verzeichnis Debug, bzw. Release , je nach Konfigurationsauswahl. Es kann nicht mir **Run** ausgeführt werden, da es ja für den ARM kompiliert ist, es müsste nun erst einmal auf den Raspberry PI kopiert werden.

Remote Debugging

Da es etwas mühsam ist nach jedem Kompilieren das Programm auf den Raspery PI zu kopieren und über SSH zu testen, ob alles so funktioniert, kann man dies alles in Eclipse integrieren. Der Raspberry PI muss hier für im Netzwerk eingebunden sein und er sollte immer die gleiche IP Adresse zugewiesen bekommen.

Zuerst müssen ein paar Plugins installiert werden. Dazu das Menü **Help → Install New Software…** wählen. Im Drop-Down-Menü **Work with:** den Eintrag “Juno – http://download.eclipse.org/releases/juno” wählen. In **Name** den Eintrag “Mobile and Device Development” wählen und die folgenden Plugins nach installieren:

- C/C++ Remote Launch
- Remote System Explorer End-User Runtime



Mit Next die Installation starten und Finish beenden, Eclipse muss neugestartet werden. Alle in dem Dialog aufgeführten Plugins sollten installiert werden.

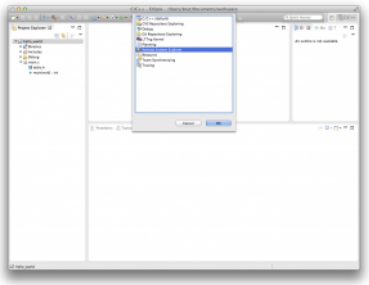
Damit der Raspberry PI auch versteht was Eclipse von ihm möchte, muss das Paket **gdbserver** auf dem Raspbery PI installiert sein. Es ist bei Raspbian zwar standardmäßig installiert, sicherheitshalber sollte es aber überprüft werden.

Des weiteren muss auf dem Raspberry PI auch ein Ort definiert sein wo das ganze, was entwickelt wird, auch gespeichert werden kann. Am sinnvollsten ist es hier die Eclipse Workspace Philosophie beizubehalten.

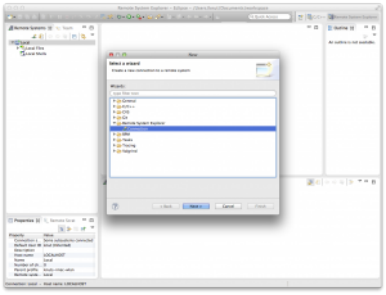
```
1 :~ $ ssh -l pi raspi
2 pi@raspi's password:
3 pi@raspi ~ $ mkdir workspace
4 pi@raspi ~ $ mkdir workspace/hello_world
```

Einrichten des Projekt für Remote Debugging

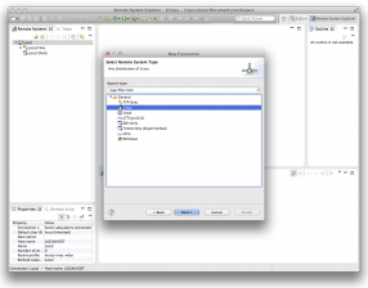
In Eclipse auf das Menü **Window → Open Perspective → Other** klicken und im Dialog **Remote System Explorer** wählen, mit **OK** bestätigen.



Anschließend im Menü auf **File → New → Other…** klicken und im Dialog unter **Remote System Explorer** den Eintrag **Connection** wählen. Mit **Next** in die nächste Dialogebene gehen.

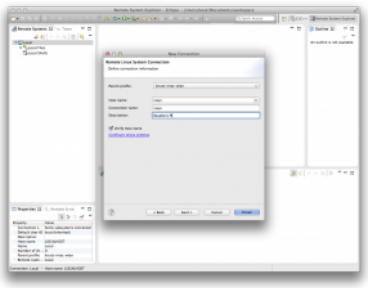


Unter **General** den Eintrag **Linux** wählen und mit **Next** in die nächste Dialogebene gehen.

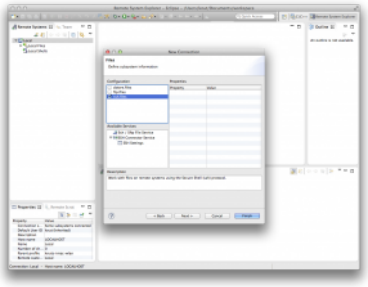


Nun müssen die Verbindungsdaten eingegebne werden.

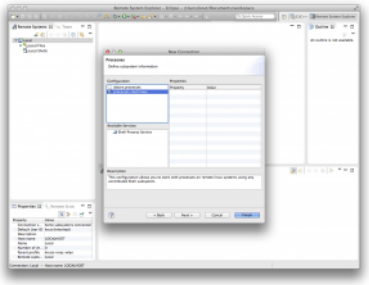
- Host Name: Die IP-Adresse des Raspberry PI oder, sofern unterstützt den Name (mein Raspberry PI heißt raspi)
- Connection Name: Ein beliebiger Name
- Description: Eine eindeutige Beschreibung der Verbindung (optional)



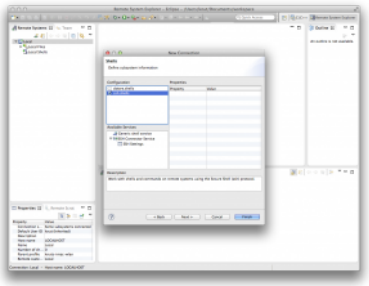
Mit **Next** in die nächste Ebene gehen, als Konfiguration “ssh.files” wählen und wieder **Next** drücken.



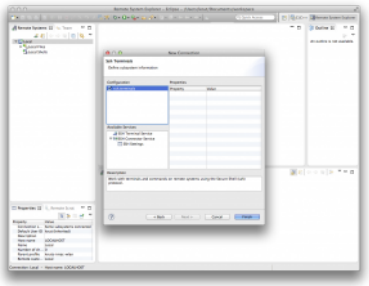
Als Prozess den Type “process.shell.linux” wählen und **Next** drücken.



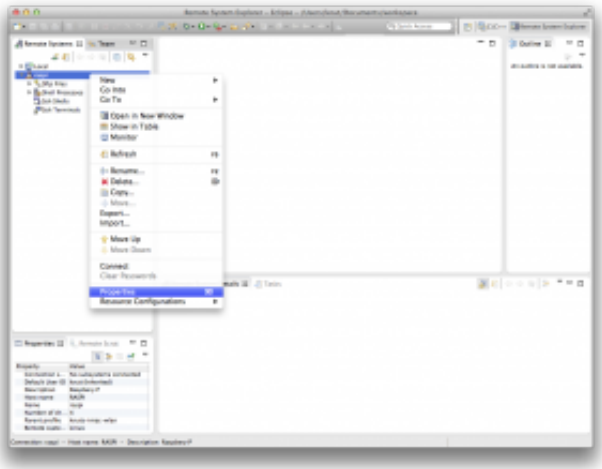
Als Shell den Type “ssh.shells” wählen und **Next** drücken.



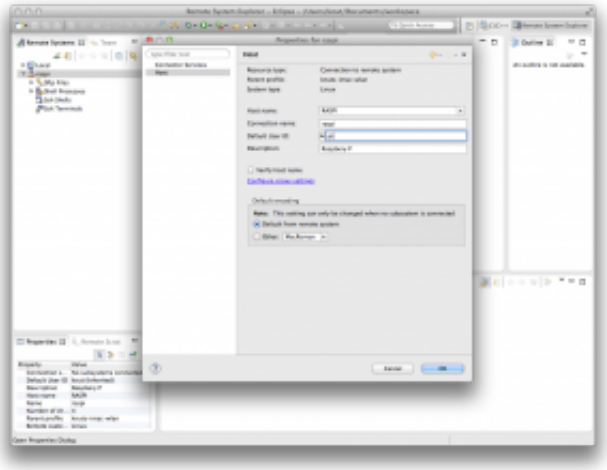
Als letzten Punkt noch das SSH-Terminal wählen und mit **Finish** abschließen.



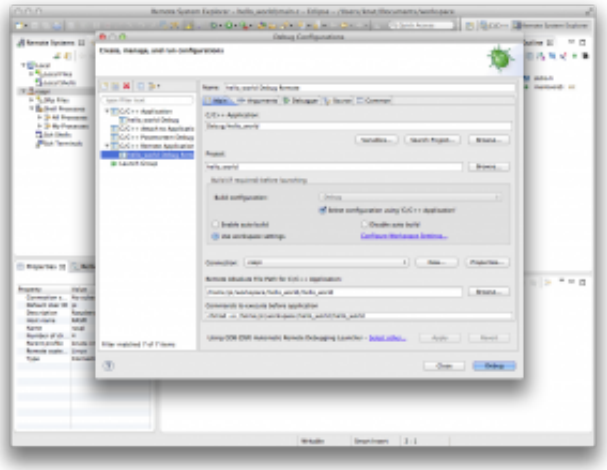
Mit einem Rechtsklick auf die Verbindung das Kontextmenü aufrufen und **Properties** wählen.



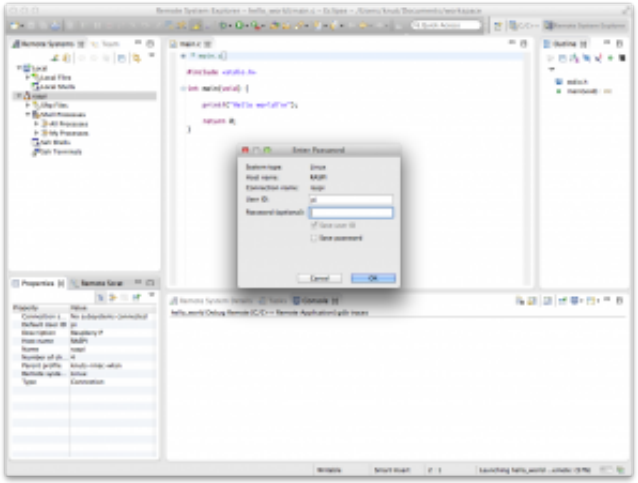
In dem Dialog, auf der linken Seite, **Host** auswählen und als **Default User ID** den Benutzernamen eingeben, zum Beispiel “pi” und mit **OK** bestätigen.



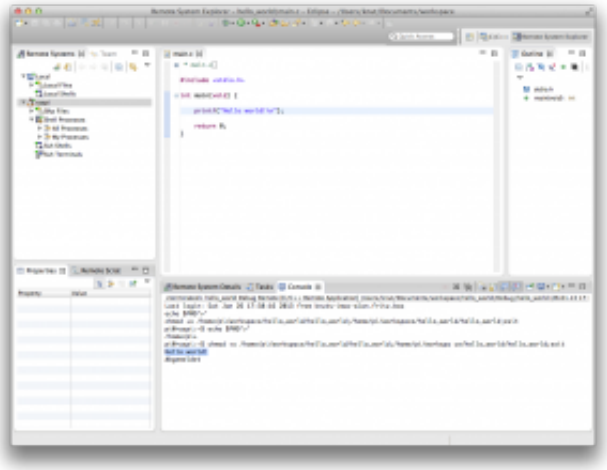
Mit dem Menü **Run → Debug Configurations** den Konfigurationsdialog aufrufen. Unter “C/C++ Remote Application” das entsprechende Projekt einbinden und den Pfad auf dem Target eingeben. Gegebenenfalls noch ein chmod execute auf die Datei setzen. Mit **Apply** die Eingaben übernehmen. Hier heißt das lokale Projekt “hello_world Debug” und das Projekt für den Raspberry PI ”hello_world Debug Remote”.



Durch Drücken des Buttons **Debug** kann die Datei nun auf dem Raspberry PI ausgeführt werden. Zuvor erscheint eine Passworteingabe, sofern man das Passwort nicht geändert hat, lautet dieses “raspberry”.



Nach der Einrichtung kann das Programm entweder über das Menü **Run → Run History → hello_world Debug Remote** oder über das Toolbar-Icon **Run** ausgeführt werden. Die Ausgabe erfolgt dann in dem Tabulator **Console**.



Mit dieser Art lassen sich bequem Anwendungen für den Raspberry PI, auch ohne weiteren Monitor und Tastatur, entwickle. Der größte Vorteil liegt aber darin, dass einem beim Kompilieren erheblich mehr Rechenleistung zur Verfügung steht. Von der Datensicherung durch TimeMaschine und der Ergonomie von Eclipse mal ganz abgesehen.

Comments

- Danke! Funktioniert super, bis auf die Geschichte mit dem Remote Debugging.

  Mein App wird schön auf den Raspberry kopiert:
  chmod +x /home/pi/workspace/hello_world/TestRaspberry;gdbserver :2345 /home/pi/workspace/hello_world/TestRaspberry;exit
  aber dann kommt eine Meldung:
  “/Users/tr/Documents/workspace/TestRaspberry/Debug/TestRaspberry”: not in executable format: File format not recognized
  Es sieht so aus, als versucht Eclipse dann doch irgendwie die Datei auf meinem Rechner zu starten, anstatt auf dem Raspberry, obwohl ich in der Konfiguration den Pfad “/home/pi/workspace/hello_world/TestRaspberry” zur executable angegeben habe.

  Hast du noch einen Tipp/eine Idee wieso?

  Tobias26. Februar 2013

- Hallo Tobias,
  das habe ich vergessen zu erwähnen. Wenn ich unter “Run” -> “Debug Configurations”, das Projekt einrichte und dann das Programm auf den Raspberry PI übertrage erhalte ich beim ersten mal die folgende Meldung:

  ```
  Error in final launch sequence
  Failed to execute MI command:
  -file-exec-and-symbols
  ```

```
/Users/knut/Documents/workspace/hello_world/Debug/hello_world
Error message from debugger back end:
"/Users/knut/Documents/workspace/hello_world/Debug/hello_world": not in executable
format: File format not recognized
"/Users/knut/Documents/workspace/hello_world/Debug/hello_world": not in executable
format: File format not recognized
```

Das ist ja auch die Meldung die bei dir ausgegeben wird.
Führe ich den Remote Debug nun erneut aus, läuft das Programm auf dem Raspberry.
Der Fehler tritt bei mir nur beim Einrichten eines neuen Projekts auf und dann nichtmehr.
Hast du nur einen Versuch unternommen oder kommt die Fehlermeldung immer wieder?

Knut26. Februar 2013

- Hallo Knut,

  ich bin deiner Anleitung gefolgt und es hat auch funktioniert.

  Die einfachen Testprogramme werden auf dem Raspi ausgeführt und die

  Terminalausgabe erscheint auf meinem Mac.

  Nun meine Frage,

  ich habe zuvor über eine Remote Desktop Verbindung von Mac auf SuseLinux unter Kdevelop4

  einige Programme geschrieben, da sich der linux server nicht in meiner Wohnung befindet ,

  und diese Programme verwenden die Boost Bibliotheken. (spez thread)

  Kannst du mir sagen ob das in dieser Konstellation auch mit eclipse und cross compiling funktioniert?

  mfg

  matze

  Matze25. März 2013

- Hi,
  Sofern du per sah auf den Server kommst und die Bibliotheken lokal verfügbar sind, sollte das möglich sein.
  Gruß

  Knut25. März 2013

Hinterlasse eine Antwort

Name
Email
Webseite

Gib den CAPTCHA Code ein!

Visuell Text

Pfad: p

Kommentar abschicken

- # Verwandte Seiten

1. Raspberry PI – Die Installation
2. Raspberry PI – Netzwerk und Wlan konfigurieren
3. Raspberry PI – Userverwaltung
4. QNAP als Subversion Server

- # Schlagwörter

307cc AirPrint Audio C/C++ DOM Google JavaScript Linux Mac MobileMe ownCloud Plugin QNAP Raspberry PI Sound SVN Tutorial Video Web Widget WordPress

- # Social

- Gefällt mir 0
- Twittern 1
- 1

Blogroll

ifun.de
» Kinderecke: “Meine erste App” gewinnt den Deutschen Computerspielpreis 2013
26. April 2013

aptgetupdateDE
» Tipp: Converto und Phlo für OS X derzeit kostenlos
26. April 2013

Apfeltalk Magazin
» Erstmals mehr Smartphones als normale Mobiltelefone verkauft
26. April 2013

bytelude
» [Quicktip] iPhone 5 Lightning Stecker hat einen Wackler
16. April 2013

apfelquak
» iPad Mini Hüllen Check + Gewinnspiel
19. Februar 2013

- **Seiten**
  - Development
    - LabVIEW
      - LabVIEW und LabVIEW RTE unter Linux
    - WordPress
      - Wel!AmazonAdds for WordPress
      - Wel!Zoom 2
  - Downloads
  - Impressum
  - Projekte
    - QNAP – Mods und Tricks
      - AFP-Symbol durch anderes ersetzen
      - Debian auf QNAP
      - Optware Init-Skripte nach Booten starten
    - Raspberry PI
      - Low Budget NAS mit einem Raspberry PI
      - Raspberry PI als AirPlay Client