Alternative HomeMatic-CCU Umgebungen im Einsatz:

Vor-/Nachteile, Inbetriebnahme und Nutzung

Jens Maus – mail@jens-maus.de

HomeMatic User-Treffen 2016, Kassel, 21. Mai 2016

Disclose Statement

- keinerlei finanzielle Unterstützung/Beziehung zu einem Drittunternehmen (z.B. eQ3)

- keinerlei Garantie auf Korrektheit bzw. technischer Unversehrtheit der eigenen Hardware/Software

- Lediglich Äußerung privater Meinungen – nicht die Meinung von eQ3, ELV oder anderer genannter Firmen/Produkte

- Erwähnte Hardwaremodifikationen können die Garantie aber auch die Zulassung von Hardware gefährden

HomeMatic Architektur

Sensoren

Direktverknüpfung

BidCos-RF / Wired / IP
(ggf. AES verschlüsselt)

Zentrale/Gateway

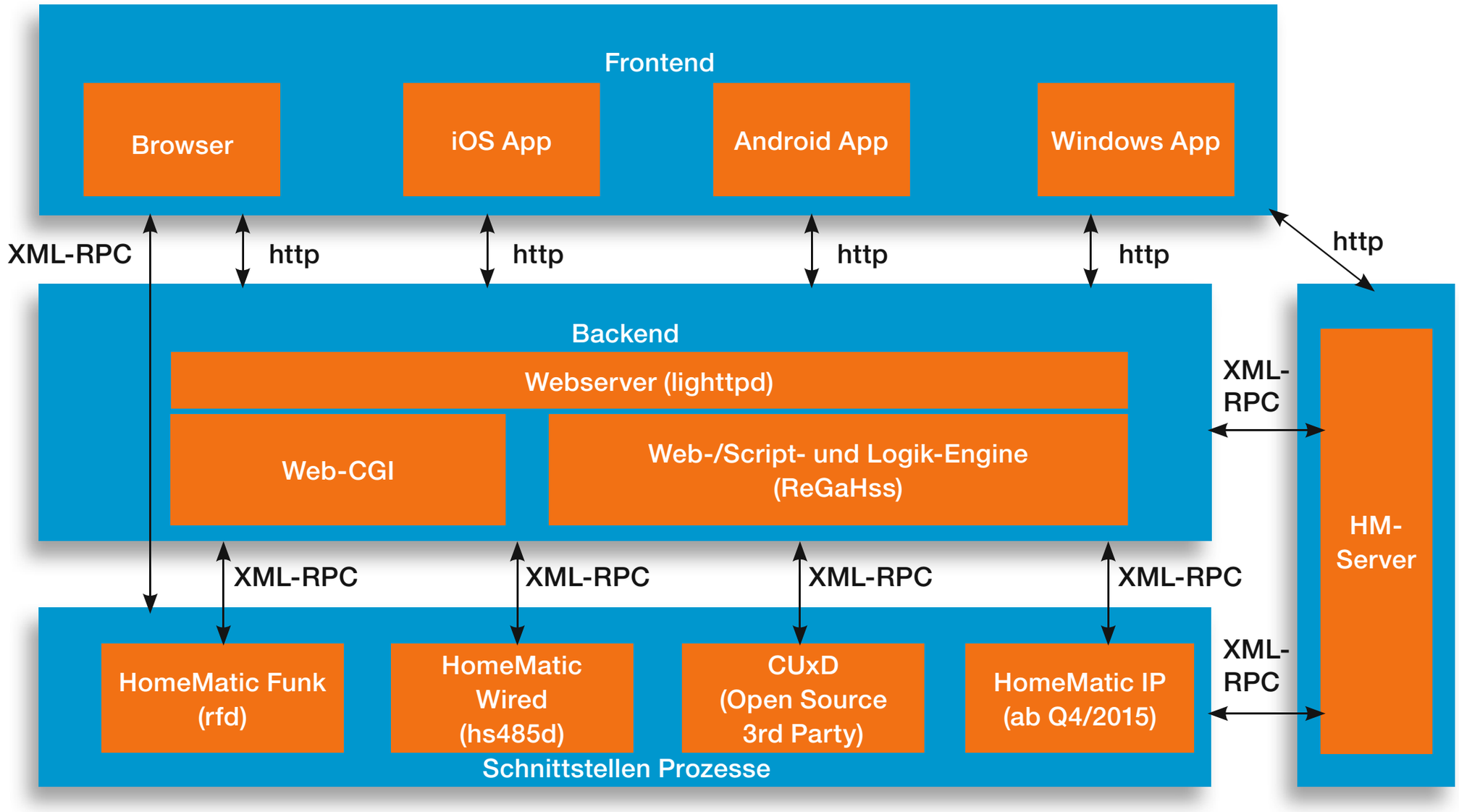
Aktoren

LAN

HomeMatic CCU

- CCU = **C**entral **C**ontrol/Communication **U**nit
- Markeinführung 2008 (CCU1) bzw. 2013 (CCU2)
- Erlaubt die zentrale Steuerung/Konfiguration von HomeMatic Komponenten mittels WebInterface (WebUI)
- Rudimentäre/grafische Programmierumgebung (inkl. properitärer Skriptsprachumgebung)
- API Schnittstellen inkl. Remote Procedure Call Funktionalität (XML-RPC, BIN-RPC) für Anbindung ext. Applikationen (z.B. CUxD, ioBroker, CCU.io, etc.)
- Addon Schnittstelle für die Erweiterung von Funktionalität durch Drittanbieter

HomeMatic CCU Softwarearchitektur



Schnittstellenbeschreibung [1]

[1] ELV Journal 05/2015

CCU-Softwarearchitektur – Vorteile

- zentralisierte Steuerung aller HomeMatic Komponenten
- rudimentäre / grafische Programmierumgebung für die Implementierung einfachster Ablaufpläne/Beziehungen zwischen HomeMatic-Komponenten (Anfängertauglich)
- einfachste Konfiguration / Einbindung in existierende LAN Strukturen
- Potentiell hohe Stabilität:
  - Betriebssystem basierend auf speziellem Embedded Linux-System (buildroot [1])
  - Hardware-Watchdog

[1] buildroot – https://buildroot.org

CCU-Softwarearchitektur – Nachteile

- Nutzung sehr heterogener Softwareumgebungen
  - rfd/hs485d/ReGaHss – C-Programme
  - HMServer – Java-Programm
  - Skripting – TCL / properitäre Skriptsprache
- Kein implementiertes Multithreading (z.B. in der Logikschicht)
- Systemkritische Dinge nicht als OpenSource veröffentlicht (z.B. ReGaHss, HM-Server) [1]
- (Funktionserweiterungen nur über Addons – kein Standard-Linux)

[1] OCCU – https://github.com/eq-3/occu

CCU-Softwarearchitektur – Nachteile

- Stabilitätsprobleme bei größeren Installationen, z.B.
  - kein Sandboxing von Addons
  - keine Realtime Kernel Nutzung (PREEMPT_RT)
  - Bugs in der Programmierschicht (ReGaHss) bringen Dienste zum Absturz
- teils gravierende Bugs/Limitationen in der Programmierumgebung (siehe später)
- (Strenge Abhängigkeit zu ARM Architektur)

CCU2 – Nachteile

- veralteter Hardwarestand
  - 32bit ARM926EJ-S
    (ARMv5 454 MHz – ~2001/2 Markeinführung)
  - 256 MB RAM
- teils sehr limitierte Funkreichweite
  (Repeater oder Antennenerweiterung [1] notwendig)

[1] http://homematic-forum.de/forum/viewtopic.php?t=16907#p136472

CCU Alternativen

CUL-Stick (868 MHz)

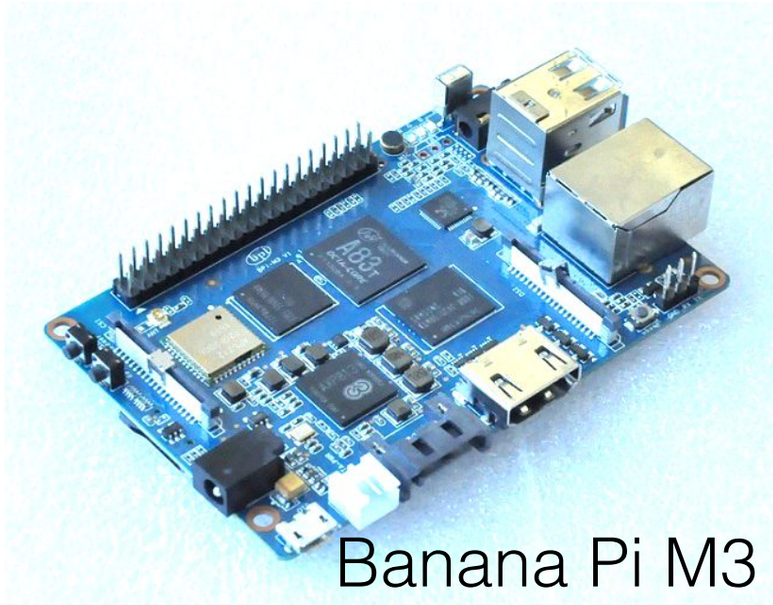
intel NUC (x86)

z.B. Fhem

Freundliche **H**ausautomatisierung und **E**nergie-**M**essung

- Hohe Flexibilität (+EnOcean, +FS20, etc.), bessere Programmiermöglichkeiten
- hohes Expertenwissen notwendig,
  keine/schwierige Firmware-Updates von HomeMatic Komponenten, etc.

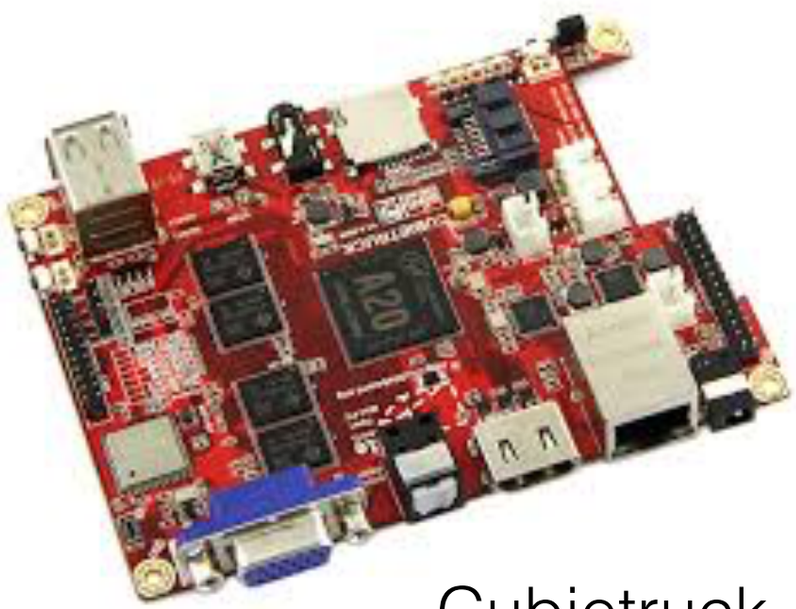
CCU2 Alternativen: ARM Einplatinenrechner



Banana Pi M3

z.B. Raspberry Pi 3:

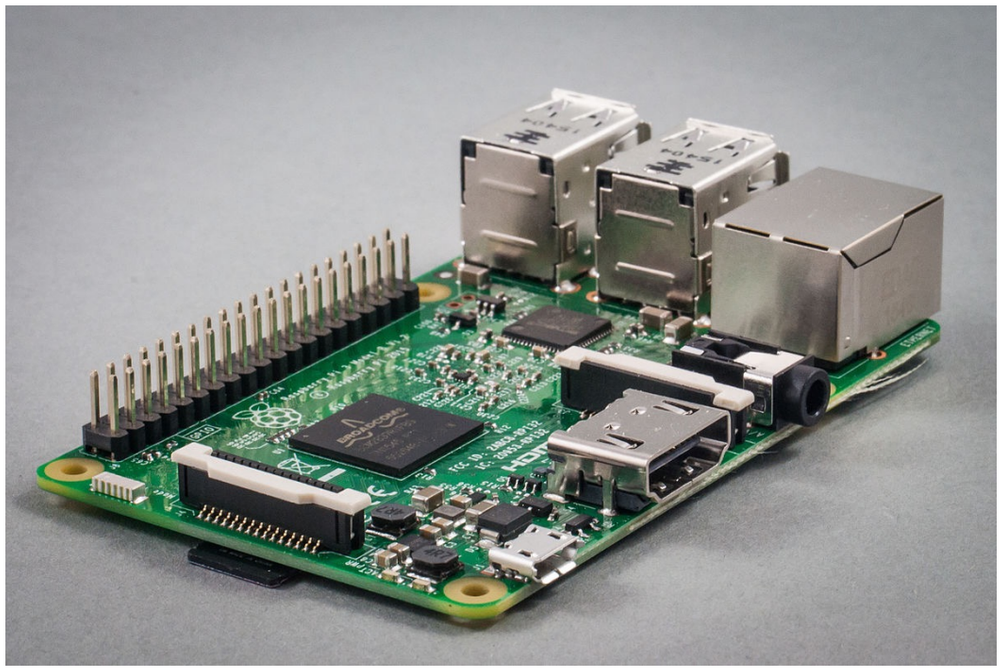
- 64Bit ARMv7 Quad Core Prozessor (1.2GHz)
  ~ 20x schneller als CCU2
- 1GB RAM
- WLAN/Bluetooth onboard



Cubietruck

CCU2 Alternativen: Funkanbindung

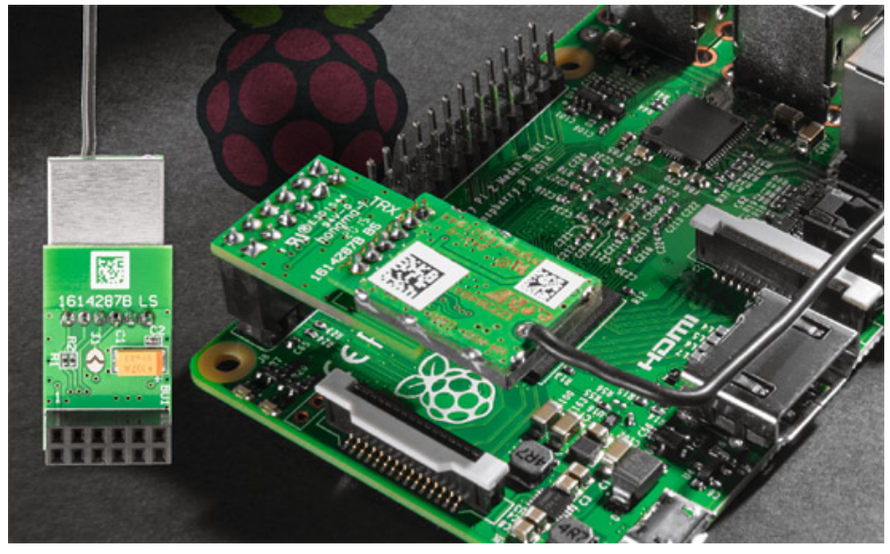
HomeMatic LAN-Gateway



Raspberry Pi 2 od. 3 (RPi)

oder

HomeMatic RPi Funkmodul



Alternative CCU Umgebungen

LXCCU
(2014)

OCCU(-SDK)
(2015)

RaspberryMatic
(2015)

YAHM
(2016)

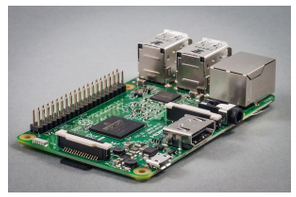
OCCU-SDK [1]

Open Central Control Unit SDK

CCU1

CCU2

ggf.



RPi

ARM/x86

Virtualisierungsansatz

LXCCU [2]

YAHM [3]

➡ 100% Community

ggf.

Distributionsansatz

RaspberryMatic [4]

➡ ~90% eQ3
~10% Community

Integrationsansatz

OCCU [5]

➡ 100% Community

[1] https://github.com/eq-3/occu
[2] http://homematic-forum.de/forum/viewtopic.php?f=26&t=18359 / www.lxccu.com
[3] http://homematic-forum.de/forum/viewtopic.php?f=18&t=31033 / https://github.com/leonsio/YAHM
[4] http://homematic-forum.de/forum/viewtopic.php?f=56&t=26917 / https://github.com/eq-3/RaspberryMatic
[5] http://homematic-forum.de/forum/viewtopic.php?f=55&t=27898

Alternative CCU Umgebungen: Integrationsansatz

OCCU – **O**pen **C**entral **C**ontrol **U**nit [1]

- Methode: Direkte Integration/Installation aller CCU Dienste (ReGaHss, rfd, WebUI, etc.) in existierendes Linux System (z.B. unter Raspbian/Debian als *.deb Pakete)

- Potential: Direkte Interaktion zwischen CCU-Architektur und Raspbian/Debian Betriebssystem möglich

- Problematik: Hohes Linux-KnowHow notwendig, Schwieriges extrahieren aller notwendigen Programme/Dienste aus OCCU-SDK; OCCU-SDK stark auf CCU2 Umgebung zugeschnitten

- Installation: Momentan nur manuelle Installation / noch kein separates Projekt verfügbar bzw. nur sehr rudimentär

[1] http://homematic-forum.de/forum/viewtopic.php?f=55&t=27898

Alternative CCU Umgebungen: Virtualisierungsansatz

LXCCU – **L**inu**X** **CCU** [1] / **YAHM** – **Y**et **A**nother **H**omematic **M**anagement [2]

- Methode: Virtualisierung (mittels Linux Container) der kompletten CCU Firmware innerhalb einer existierenden Linux-Umgebung (z.B. unter Raspbian/Debian) mit Schnittstellenanpassungen an das Host-OS

- Potential: Sehr updatefreundlich (offizielle eQ3 CCU Firmware Releases nutzbar); klare Trennung zwischen CCU Umgebung und Raspbian/Debian-System; potentielle Unterstützung anderer Linux-Derivate; Nutzung aller Addons ohne Anpassungen

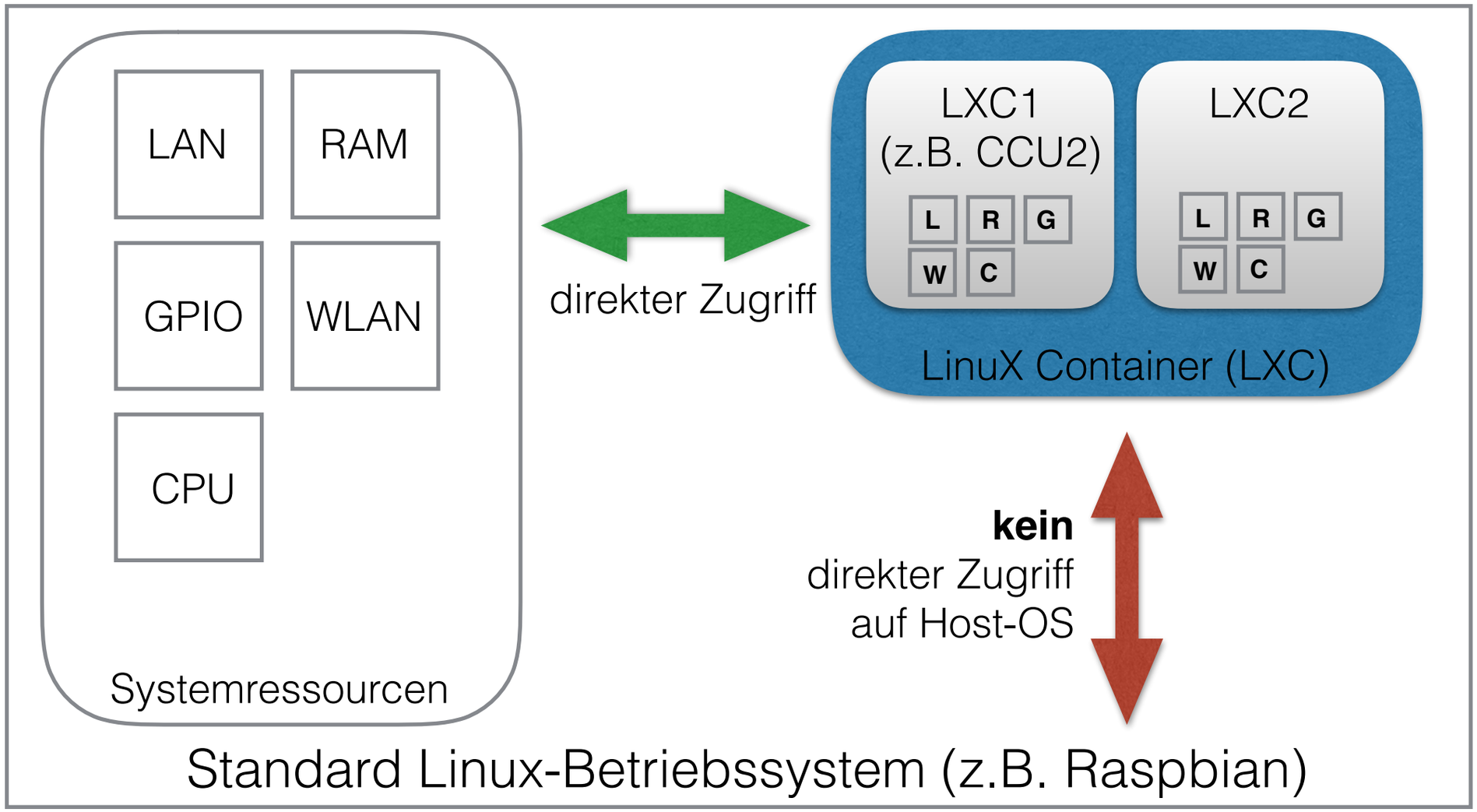
- Problematik: Mittleres Linux-KnowHow (Kommandozeile) notwendig; Raspbian/Debian Updates mit Vorsicht zu genießen; direkte Nutzung von CCU Firmwares problematisch

- Installation: Einfach; Aktuelle CCU Firmwares (2.19.9) verfügbar (YAHM)

[1] http://homematic-forum.de/forum/viewtopic.php?f=26&t=18359 / www.lxccu.com
[2] http://homematic-forum.de/forum/viewtopic.php?f=18&t=31033 / https://github.com/leonsio/YAHM

Alternative CCU Umgebungen: Virtualisierungsansatz

LXCCU – **L**inu**X** **CCU** [1] / **YAHM** – **Y**et **A**nother **H**omematic **M**anagement [2]



Alternative CCU Umgebungen: Virtualisierungsansatz

<u>Vergleich LXCCU vs. YAHM:</u>

- **LXCCU**
  - Basiert auf Installation von vorbereiteten Debian/Raspbian Paketen (*.deb)
- **YAHM**
  - Installation/Konfiguration mittels Kommandozeilen-Skripten
  - Linux-Derivat unabhängig (z.B. auch Ubuntu/Armbian, etc.)
  - Auf für x86 Linux-Systeme geplant

Alternative CCU Umgebungen: Virtualisierungsansatz

Installationsablauf (prinzipiell)

1) Grundinstallation Raspbian/Debian Linux

2) Installation der Virtualisierungslösung

- **LXCCU**

```
wget -nv -O- http://www.lxccu.com/setup.sh | sudo bash -
```

RPi-Funkmodul:

```
wget http://cdn.lxccu.com/hm-mod-rpi-pcb_lxccu.sh ; chmod +x hm-mod-rpi-pcb_lxccu.sh
./hm-mod-rpi-pcb_lxccu.sh install
```

- **YAHM**

```
wget -nv -O- https://raw.githubusercontent.com/leonsio/YAHM/master/yahm-init | sudo -E  bash -s
quickinstall -
```

3) Reboot

4) Zugriff auf gewohnte CCU2 WebUI Oberfläche
http://homematic-ccu2/

Alternative CCU Umgebungen: Distributionsansatz

RaspberryMatic [1]

- Methode: Distribution einer speziell für RaspberryPi 2 und 3 angepassten CCU Firmware mit direkter Unterstützung des RPi-Funkmoduls

- Potential: Simple Installation (SD Kartenimage + Funkmodul); direkte Verteilung von Updates durch eQ3; Potential als künftige CCU Referenzplatform; Potentiell hohe Stabilität (nur ein OS-Level)

- Problematik: Immer noch Beta-Status (finale version zugesagt für Q1/2016); Firmware-Stand noch bei 2.15.5; Datenübernahme von CCU2 semi-automatisch; Zusatzsoftware (ioBroker, etc.) nur über Addon-Schnittstelle möglich

- Installation: Einfache Installation selbst für Linux/Raspberry-Einsteiger

[1] http://homematic-forum.de/forum/viewtopic.php?f=56&t=26917 / https://github.com/eq-3/RaspberryMatic

Alternative CCU Umgebungen: Distributionsansatz

RaspberryMatic

Installationsablauf (prinzipiell)

1) Download & Installation des SD-Kartenimage mittels DiskImage-Utility

2) ggf. manuelle Anpassungen für RaspberryPi3

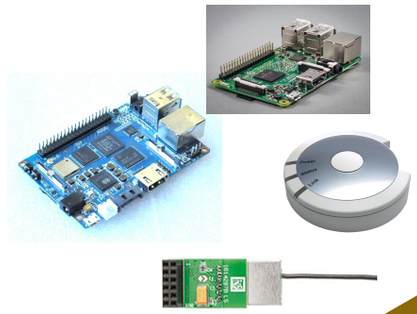
3) Initialer Start und Anpassungen (NTP, SSH)

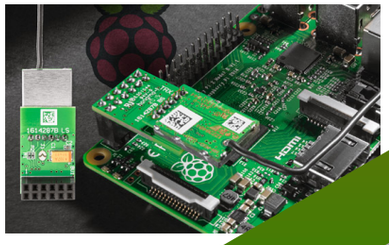
4) ggf. Datenübernahme von CCU2
   (manuelles Entfernen der CCU2-Addons vorher notwendig)

5) Installation Java-Addon (für HMServer)

6) Installation benötigter Addons (z.B. CUxD, XML-API, etc.)

CCU Umgebungen: Welche?





Komplexität

Flexibilität

?CCU3

OCCU
➡Linux-Entwickler
\+ Max. Flexib.
\- Hohe Komplex.

?CCU3

LXCCU / YAHM
➡Linux-Experte
\+ Paralleler Betrieb Raspbian
\- Stabilität/Resourcen?

?CCU3

RaspberryMatic
➡Linux/RPi-Anfänger
\+ Einfachheit / Potential
\- Updates?

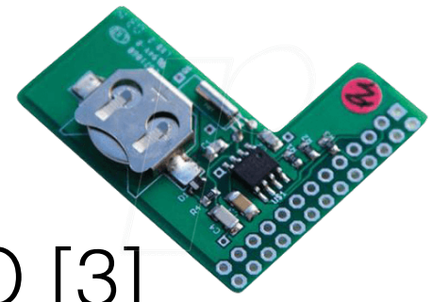
CCU2
➡HomeMatic-Einsteiger
\- Auslaufmodell

Alternative CCU Umgebungen: Probleme/Limitationen

- Wie bei CCU2: Schlechte Empfangs/Sendequalität mit Funkmodul

➡ Tausch der „Stummelantenne“ gegen leistungsstarken Ersatz [1,2]

- RaspberryPi hat keine RTC-Clock



➡ Einsatz eines zusätzlichen RTC-Modules am GPIO [3]

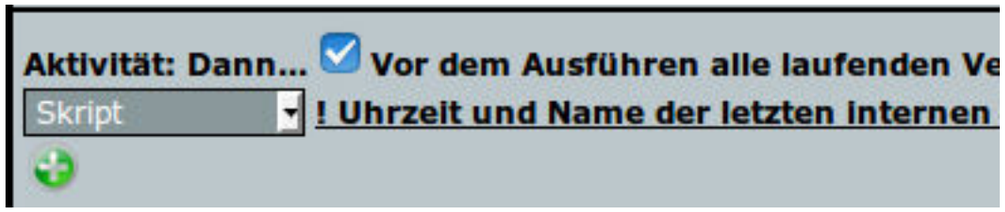
- Keine standardmäßige meine-homematic.de Unterstützung (für RaspberryMatic existiert Addon [4])

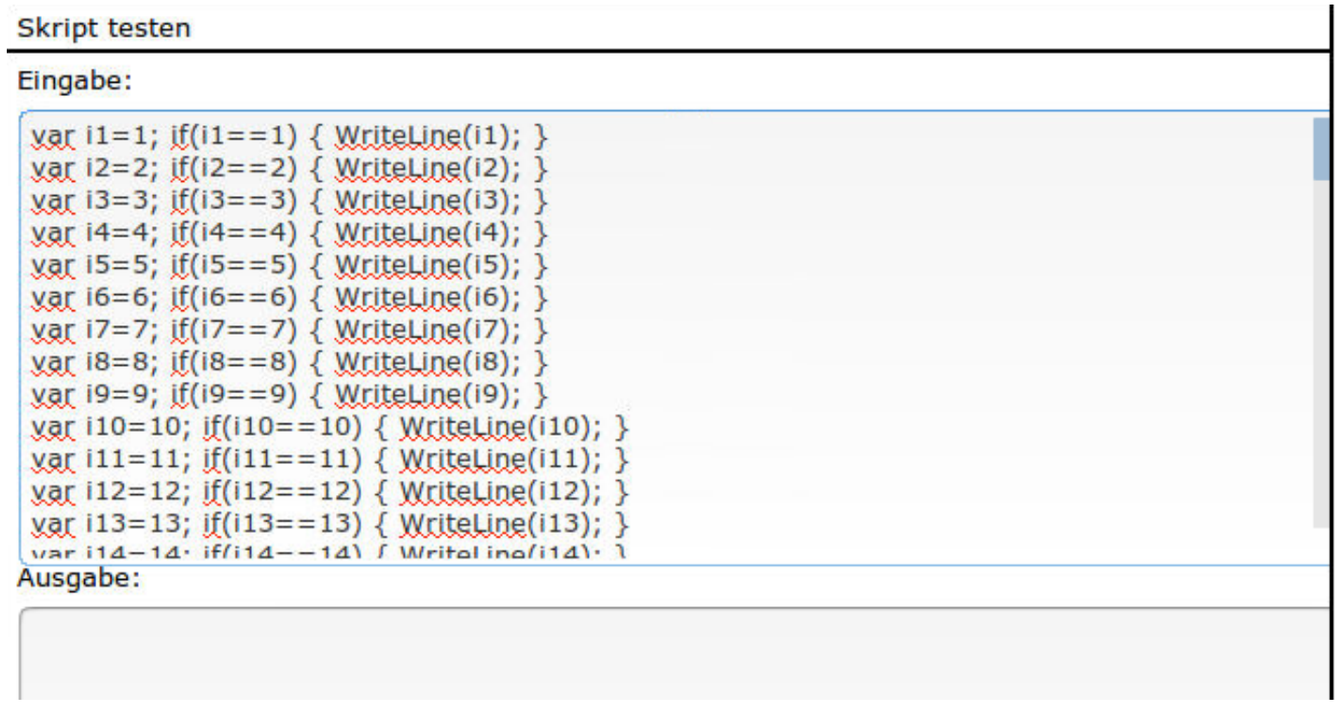
[1] http://www.stall.biz/project/externe-antenne-stabantenne-fuer-raspberrymatic-occu-hm-mod-rpi-pcb
[2] http://homematic-forum.de/forum/viewtopic.php?f=27&t=27287
[3] http://homematic-forum.de/forum/viewtopic.php?f=56&t=27684
[4] http://homematic-forum.de/forum/viewtopic.php?f=56&t=26917&start=820#p279362

Alternative CCU Umgebungen: Probleme/Limitationen

- Limitationen bei Skriptprogrammierung mit modernerer Hardware noch problematischer als mit CCU2

(1) max. 200 Skriptvariablennamen (Systemweit)





➡ keine Ausgabe bzw. Fehler bei >200 unterschiedlichen Variablennamen

➡ d.h. nicht mehr als 200 Variablen CCU-weit einsetzbar

➡ beeinträgt eigene Skriptentwicklung sehr

Alternative CCU Umgebungen: Probleme/Limitationen

- Limitationen bei Skriptprogrammierung mit modernerer Hardware noch problematischer als mit CCU2

  (2) Skriptvariablen verlieren am Ende nicht ihre Gültigkeit

  ➡ systemkritische Wiederverwendung von Variablenwerten in zwei unterschiedlichen Programmen mit schwer erkennbaren Effekten

  (3) 50kB Limit für Skripte

➡ Bei komplexer Skriptprogrammierung Umstieg auf alternative Lösungen (ioBroker, pmatic, pyhomematic) unausweichlich

„IMHO“ / Wünsche an eQ3

(1) größeres Commitment bzgl. RaspberryMatic

- zeitnah finale Version mit aktueller Firmware (2.19.9)

- regelmäßige Firmware Updates synchron mit CCU2 (unified build system?)

- Update auf aktuellste Buildroot-Umgebung (2016.02) ggf. sogar mit RealTime Support (PREEMPT_RT)

- In Zukunft als Referenzplatform nutzen / propagieren

„IMHO“ / Wünsche an eQ3

(2) größeres OpenSource-Engagement (OCCU-SDK)

- direkte Entwicklungsarbeit in Github (nicht nur Snapshots)
- Übernahme von Änderungen in Endprodukt die mittels Pullrequests von Dritten übermittelt werden.
- Nutzung öffentlicher Bug/Issuetracker
- Veröffentlichung komplettes Buildsystem
  (z.B. eigene Generierung von RaspberryMatic Images bzw. sogar mitunter komplette CCU2 Firmware Image)
- Veröffentlichung weiterer Quellcodes (z.B. ReGaHss) – Lizenzproblematik müsste geklärt werden (Crowdfunding?)

„IMHO“ bzw. Wünsche an eQ3

(3) stärkere Konzentration auf Kernkompetenzen

- Fokus auf mehr/bessere Aktoren/Sensoren/Gateways
- Statt ggf. CCU3-Hardwareentwicklung Konzentration auf RaspberryPi-basierte Referenzplatform (ggf. OEM-Partnern die Entwicklung+Vertrieb von Komplettsystemen überlassen)
- Fokus auf CCU-Firmware als reine Referenzarchitektur Konzentration auf rfd, hs485d als Kernkomponenten
- Bekannte Bugs/Limitationen in ReGaHss beseitigen (ggf. auf die OpenSource Community zurückgreifen)

Was kann/sollte die Community beitragen?

- Weiterhin viel eigene (Software)entwicklungen rund um das Thema HomeMatic

- Mehr Anleitungen/Tutorials/HowTos auf homematic-forum.de um mehr Anfängern den Einstieg zu erleichtern

- Mehr OpenSource Engagement (z.B. via Github) und Softwareentwicklung die allen auch unentgeltlich zugute kommt (siehe Smartphone App Angebot)

- Direkte Beteiligung an der Weiterentwicklung von OCCU-SDK um eQ3 mehr Motivation für weitere Offenlegungen von Quellcode zu geben