Hardware Hacking 101

Christopher Scheuring

37C3 – Lightning-Talk

/whoami

- Christopher „ChrisS“ (Scheuring) <chris+ccc@aucmail.de>
  - Hacker, Security Researcher & Analyst
  - Offiziell Security Experte (IT/OT/Embedded) und freiberuflicher Dozent (DHBW Mosbach und Karlsruhe)
- Mehr auf der Offensive-Seite… mit starkem Fokus auf Secure Development und Design
  - Hauptfokus: IoT / OT+ICS / Embedded / Multi-Tier-Umgebungen / Netzwerk
  - Ebenso Automotive, Mobile und SDR

Ziele und Motivation

- Das ganze ist als Idee für eine IT-Security Vorlesung an der DHBW entstanden
  - Zielgruppe: Informatiker, Angewandte Informatik, Embedded Entwickler
- „Einfaches“ Erkennen von Schnittstellen und Schwachstellen – daher einfach gehalten!
  - Hardware-Design
  - Software-Implementierung
- Warum?
  - Aus Security-Sicht muss auch Hardware betrachtet werden (neben phy. Zugriff...)
  - Schlechtes Software-Design dazu kann zu erfolgreichem Hack führen
    - Race-Conditions, Fehler im Exception-Handling, versteckte Funktionen...

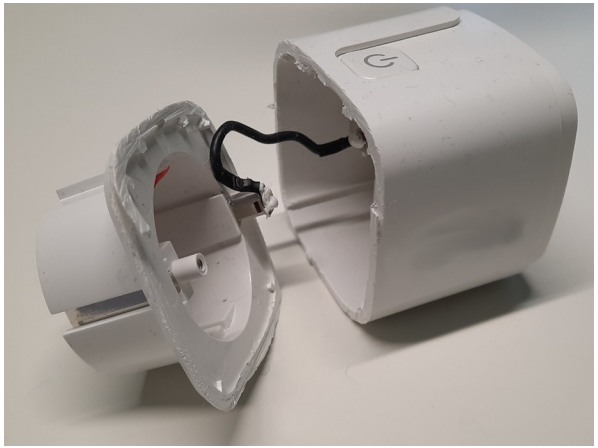
Wieso ein Hardware Hacking 101?

- Embedded und IoT Hardware wird immer günstiger und einfach zu Programmieren
  - (Meist) kein Assembler notwendig - Programmierung in LUA, MicroPython, C/C++
  - Arduino, ESP usw.
- Kaum jemand macht sich Gedanken über Security
  - Oder: Viele Entwickler denken, dass ja keiner der Hardware öffnet und/oder ist ja nur ein einfaches Stück Hardware für eine dedizierte Anwendungen…
- Die Hacking-Hardware ist teuer -.- (es war einmal…)
- Was kann schon schief gehen – ist doch verklebt oder sogar vergossen ;-)

„How to“: Öffnen von Dingen :-)

Gummihammer: Sehr zuverlässig bei geklebten Gehäusen - und richtet (meist) wenig Schaden an :-)

Wenn der Gummihammer nicht funktioniert - dann geht es mit ein wenig “Gewalt” aka Mini-Flex / Puksäge / Mini-Fräse...



Für die ganz widerspenstigen Dinge:

Am Ende geht es um eine Platine :-)

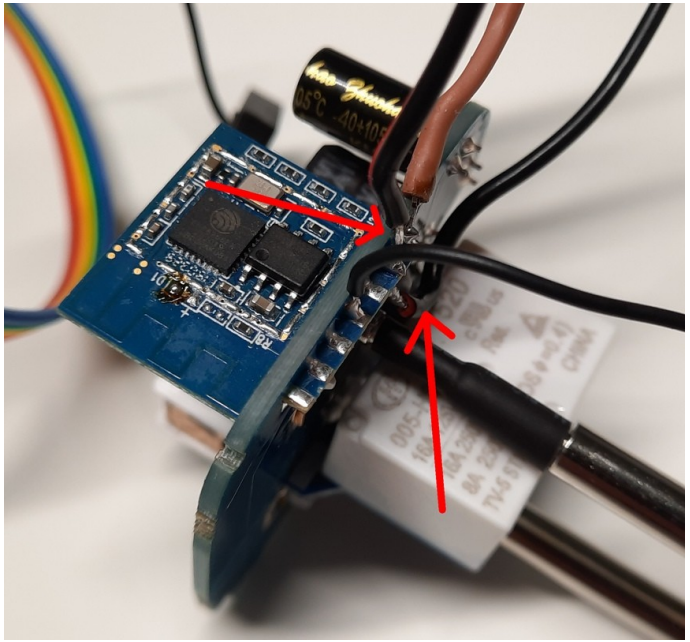
Wo Kabel angelötet werden können...

Firmware usw. ausgelesen werden kann...

Um z.B. an Credentials zu gelangen...

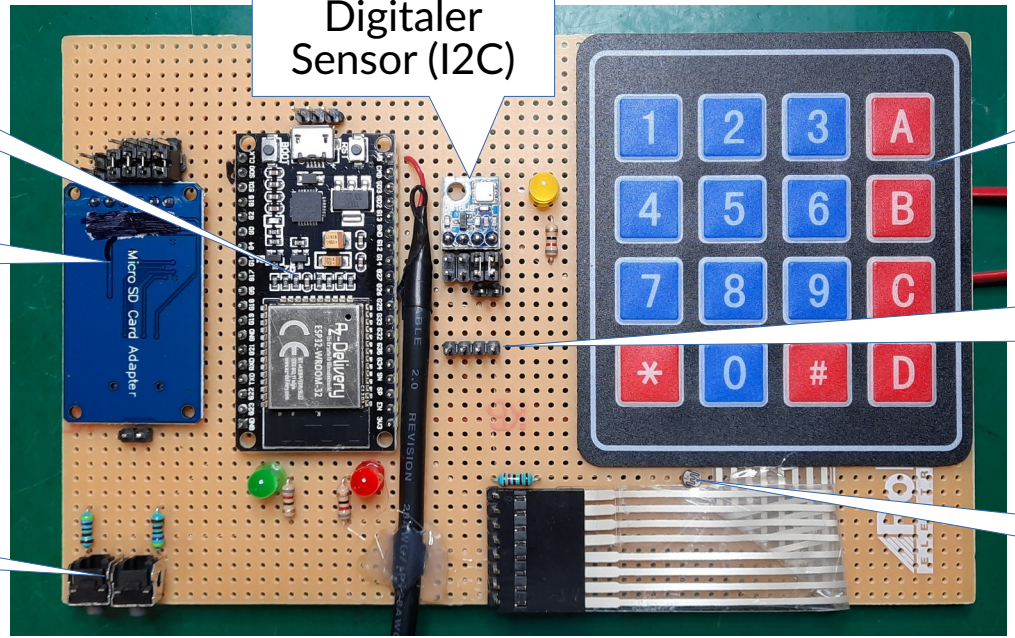
Oder die Funktion zu verstehen...

Oder eigene Firmware aufzuspielen...



Die Hardware

Das Test-Board und seine Komponenten



ESP32 WROOM-32

Digitaler Sensor (I2C)

4x4 Key-Pad

“Flash” Speicher (SPI)

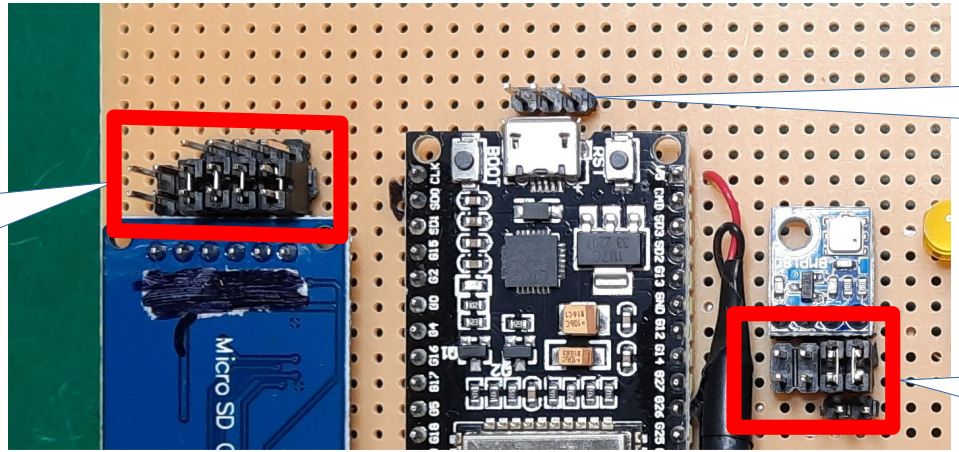
UART/TTY nur TX-Data

Taster

Analoger Licht-Sensor

Das Test-Board und seine Komponenten

Pin-Header
mit Jumper:
SPI-Bus
Vcc
GND



Keine Funktion
nur „Physical
Security“ ;-)

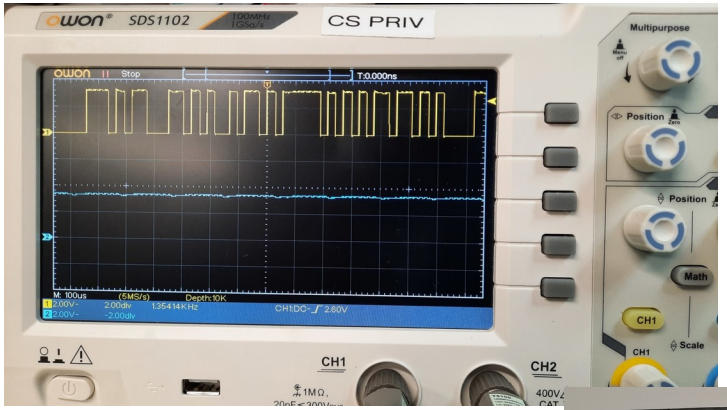
Pin-Header
mit Jumper
I2C-Bus
Vcc
GND

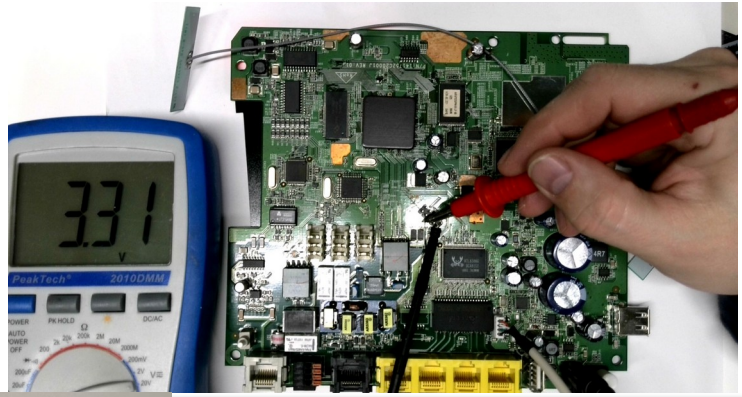
Jumper und Pin-Header dienen:

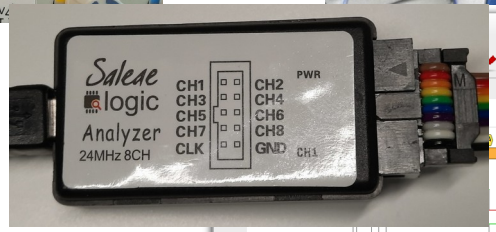
- zum besseren Zugriff zum Messen und analysieren der Signale
- zur Simulation der Unterbrechung von CLK, CS, Data-Lines usw.

Die Challenges

„How to” Multimeter und Co.







Hardware- und Funktionsanalyse

- Systematische Analyse der Funktion der Anwendung und Hardware
- Fragen
  - Welches Verhalten kann beim Ziehen einzelner Jumper identifiziert werden?
  - Welche Funktionen haben die verschiedenen Sensoren, Taster, Pin-Headers usw.?
  - Gibt es Race-Conditions, die zu unerwartetem Fehlverhalten führen?
  - Gibt es versteckte Funktionen?
- Debug- und Konsolenausgaben analysieren

Ausblick / Wo finde ich mehr dazu?

- Pläne:
  - Re-Design für flexibles PCB mit einfacher Austausch von Komponenten
  - Neue Challenges - irgendwann finden die Studies bestimmt mein Gitea ;-)
- Details zur Hardware, Sourcecode, mehr Doku, Übungen usw:
  - https://gitfoo.0x17sec.de/DHBW-Stuff/hw-hacking-101
  - soweit möglich Unter GPL und CC veröffentlicht
- Oder direkt an mich ;-)
  - chris+ccc@aucmail.de
  - @0x4045494650@chaos.social :: @0x4045494650