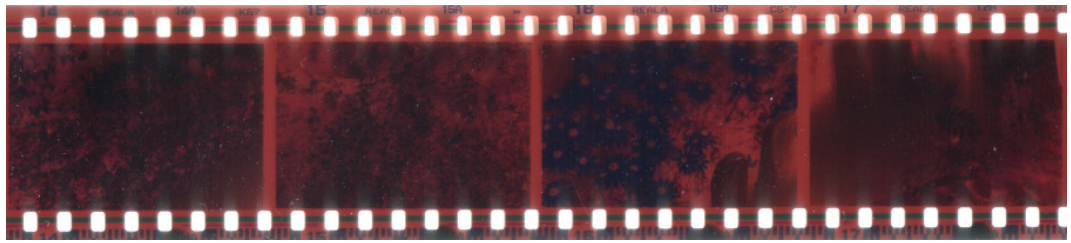
Entwicklung eines Treibers für einen Filmscanner mittels USB-Sniffing und Reverse Engineering

Hugo Platzer

Reflecta CrystalScan 7200

- Zum Digitalisieren von Kleinbilddias / -negativen



- Per USB verbunden
- Software nur für Windows / Mac
- Versuch, einen Linux-Treiber zu entwickeln ohne entsprechende Dokumentation

Reverse Engineering

- Prozess des „Engineering“ (von der Spezifikation zum Produkt) in umgekehrter Richtung
- Versuch, aus für den Konsumenten verfügbaren Daten (Firmware, Software, Mitlauschen an Schnittstellen) Protokolle zu rekonstruieren
- Rechtliche Grauzone
- Um Einverständnis gebeten: Hersteller war kooperativ

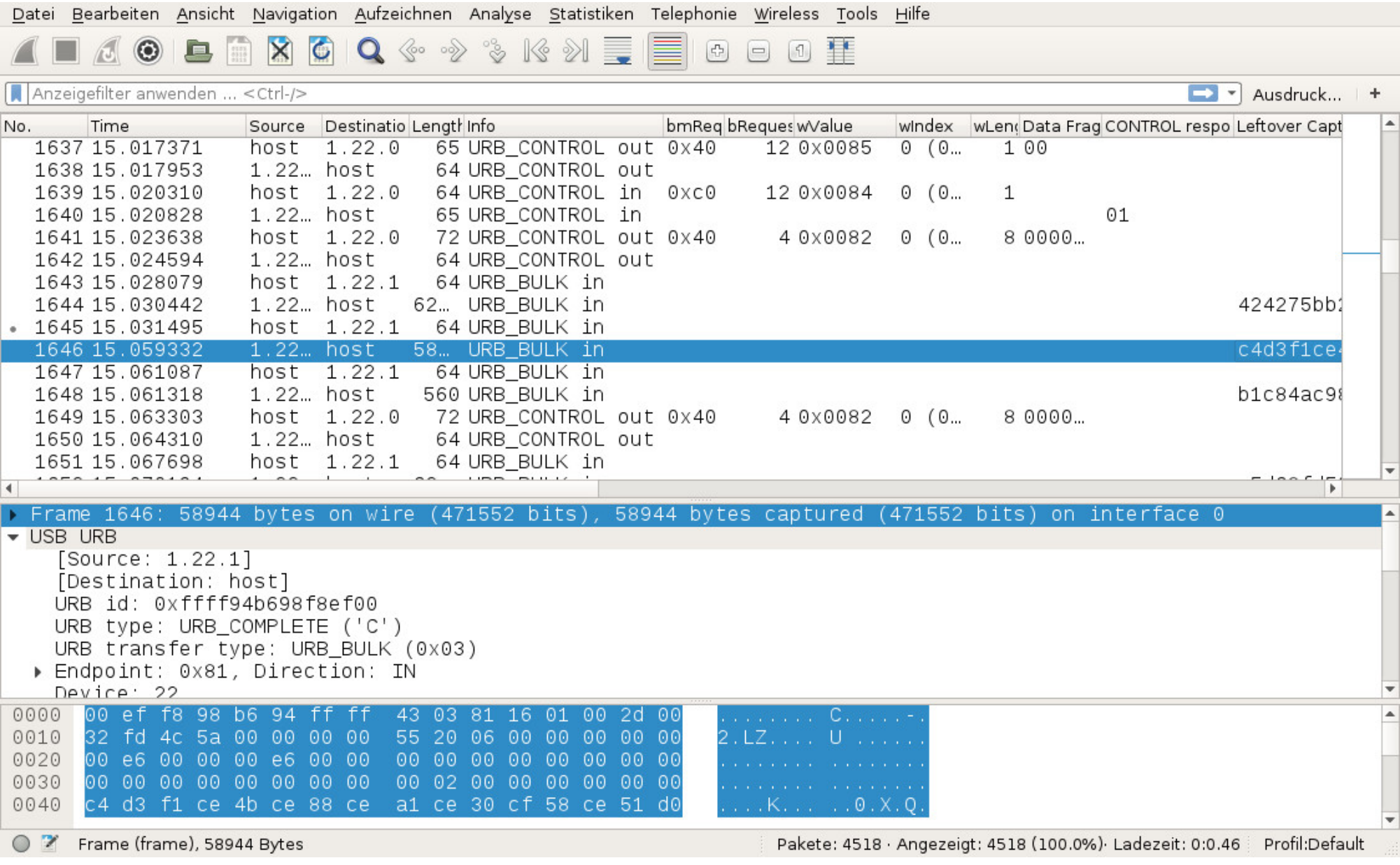
Der USB-Standard

- ein Anschluss für alle Klassen von Geräten
- Alle Kommunikation geht vom Host aus
- Übertragung in Paketen > Transactions > Transfers
- ein Gerät hat oft mehrere Kanäle (Endpoints)
- Control-Endpoint
  - besitzt jedes Gerät, dient zur Steuerung
  - definierte Nachrichtenform mit Parametern und Zusatzdaten
  - Standard Requests: Vom USB-Standard vorgeschriebene Basisfunktionen
  - Vendor-specific Requests: Herstellerspezifische Zusatzfunktionen, undokumentiert
- Bulk-Endpoint
  - für große Datenmengen (Dateien, Bilddaten)
  - Bytestrom ohne Struktur

USB-Sniffing

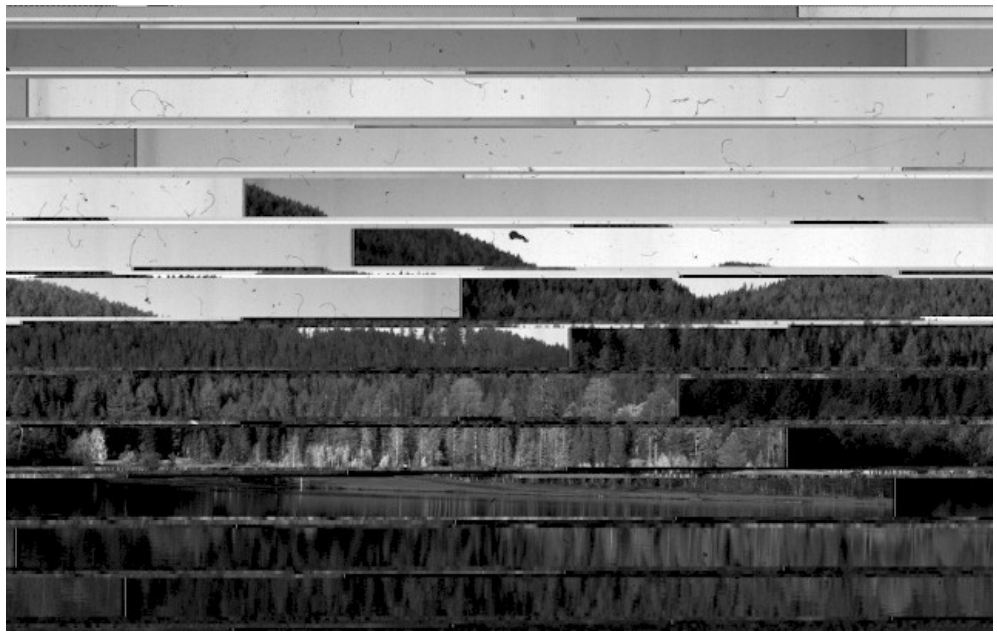
- Mitschneiden der Kommunikation zwischen Gerät und Windows-Software
- kann in Software oder Hardware umgesetzt werden
- Kernelmodul `usbmon`
  - Zugang zu vom Linux-Kernel abgearbeiteten USB-Transfers
  - Schwierigkeiten bei langer Payload
- Wireshark
  - Netzwerk-Sniffer, der sich auch für USB eignet
  - sowohl zur Aufzeichnung als auch Analyse
  - flexibel anpassbare Oberfläche
- VirtualBox
  - Windows in virtueller Maschine unter Linux ausführen
  - USB-Passthrough gibt Windows Zugang zum Scanner

Wireshark


Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telephonie Wireless Tools Hilfe
Anzeigefilter anwenden … <Ctrl-/>
Ausdruck…
No. Time Source Destinatio Lengt Info bmReq bReques wValue wIndex wLen Data Frag CONTROL respo Leftover Capt
1637 15.017371 host 1.22.0 65 URB_CONTROL out 0x40 12 0x0085 0 (0… 1 00
1638 15.017953 1.22… host 64 URB_CONTROL out
1639 15.020310 host 1.22.0 64 URB_CONTROL in 0xc0 12 0x0084 0 (0… 1
1640 15.020828 1.22… host 65 URB_CONTROL in 01
1641 15.023638 host 1.22.0 72 URB_CONTROL out 0x40 4 0x0082 0 (0… 8 0000…
1642 15.024594 1.22… host 64 URB_CONTROL out
1643 15.028079 host 1.22.1 64 URB_BULK in
1644 15.030442 1.22… host 62… URB_BULK in 424275bb
1645 15.031495 host 1.22.1 64 URB_BULK in
1646 15.059332 1.22… host 58… URB_BULK in c4d3f1ce
1647 15.061087 host 1.22.1 64 URB_BULK in
1648 15.061318 1.22… host 560 URB_BULK in b1c84ac9
1649 15.063303 host 1.22.0 72 URB_CONTROL out 0x40 4 0x0082 0 (0… 8 0000…
1650 15.064310 1.22… host 64 URB_CONTROL out
1651 15.067698 host 1.22.1 64 URB_BULK in
Frame 1646: 58944 bytes on wire (471552 bits), 58944 bytes captured (471552 bits) on interface 0
USB URB
[Source: 1.22.1]
[Destination: host]
URB id: 0xffff94b698f8ef00
URB type: URB_COMPLETE ('C')
URB transfer type: URB_BULK (0x03)
Endpoint: 0x81, Direction: IN
Device: 22
0000 00 ef f8 98 b6 94 ff ff 43 03 81 16 01 00 2d 00 .... C.....-.
0010 32 fd 4c 5a 00 00 00 00 55 20 06 00 00 00 00 00 2.LZ.... U ......
0020 00 e6 00 00 00 e6 00 00 00 00 00 00 00 00 00 00 ....
0030 00 00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 ....
0040 c4 d3 f1 ce 4b ce 88 ce a1 ce 30 cf 58 ce 51 d0 ....K... ..0.X.Q.
Frame (frame), 58944 Bytes
Pakete: 4518 · Angezeigt: 4518 (100.0%)· Ladezeit: 0:0.46
Profil:Default

usbmon: unvollständige Payload

- Bei großen Paketen wird Payload nach 61440 Bytes abgeschnitten
- visuell: Risse im rekonstruierten Bild



- gelöst durch einfachen Kernel-Patch

```
if (length >= rp->b_size / 5) length = rp->b_size / 5;
```

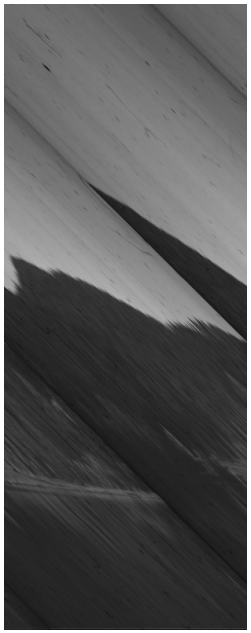
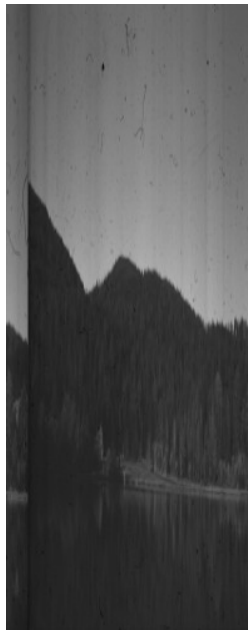
Analyse der Aufzeichnung

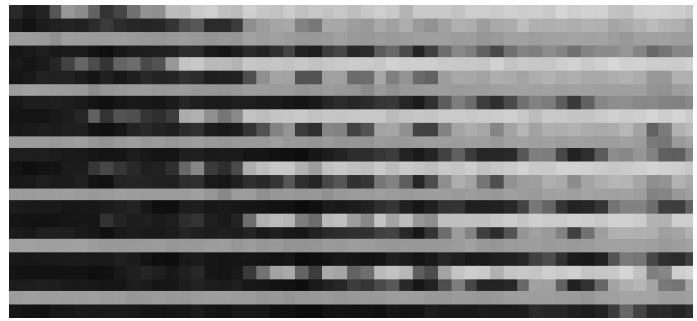
- Versuche, das übertragene Bild aus Aufzeichnung zu rekonstruieren
- Parsen des .pcap-File mit `tshark`
  - in Wireshark alle Protokolle außer USB deaktivieren!
- Bilddaten werden über große Bulk-Transfers übertragen
- zeilenweise 16-bit Little Endian Pixelwerte

```
40  c8 3b 5d 31 ae 2a 78 29  1c 29 bd 2e e9 33 93 40
50  21 43 25 2a 77 1d 76 1e  98 28 25 29 41 22 63 26
60  50 22 4a 1d 41 18 58 18  cf 17 9e 1a d3 1c 8b 1d
70  94 21 48 1e a8 24 10 1f  1f 15 78 15 15 1a 09 1b
```

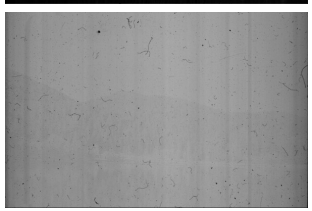
- eingehende Bytes am Bulk-Endpoint sammeln, dann weiterverarbeiten:
  - Offsets
  - $16 \rightarrow 8$ Bit
  - In Zeilen gruppieren
  - usw.

Rekonstruktion des Bildes





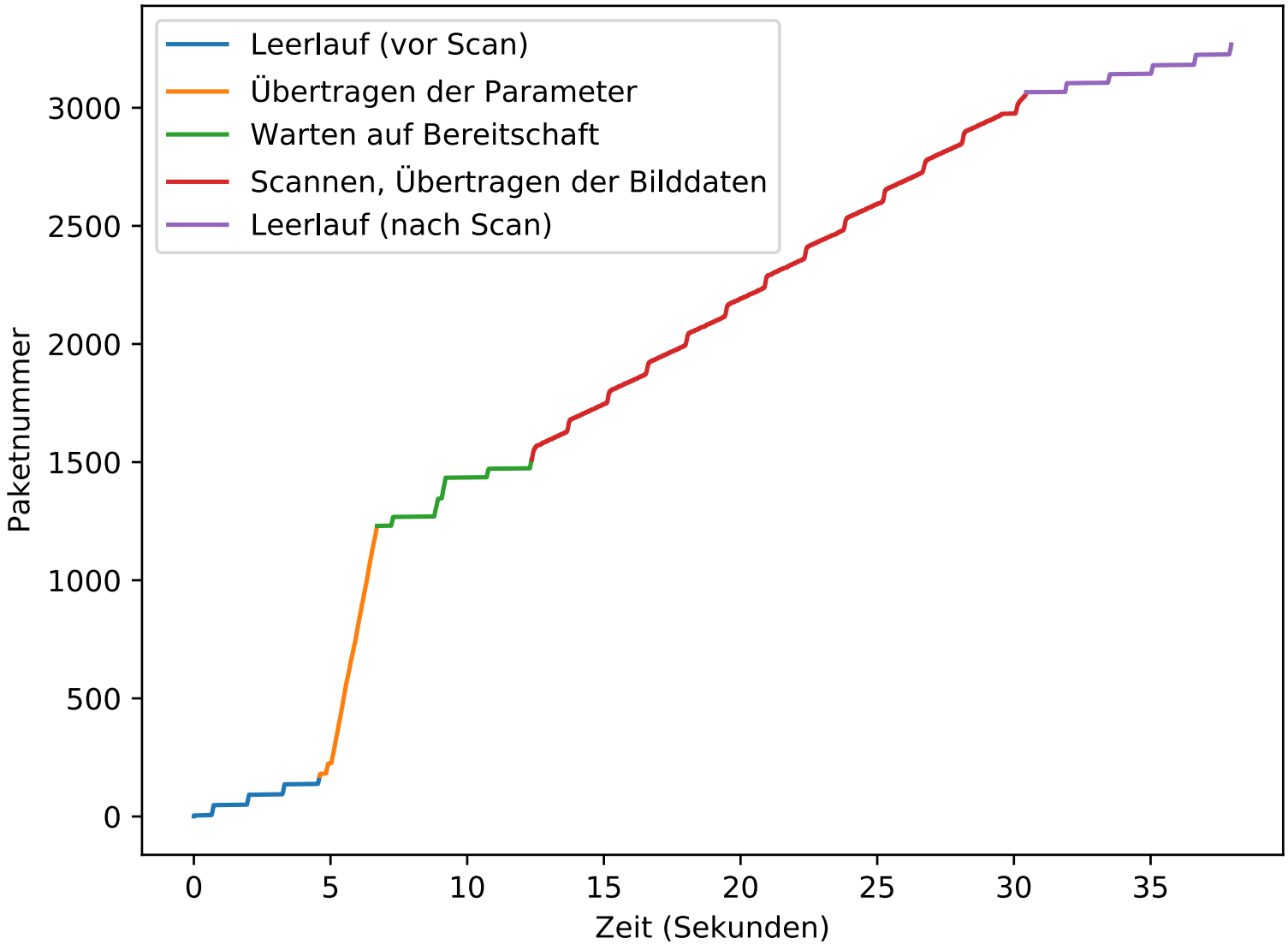



Ansteuern des Scanners

- PyUSB - basiert
- Grundidee: aufgezeichnete Befehlsfolge zum Gerät wieder abspielen, eingehende Daten speichern
- viele Muster in der Kommunikation erkennbar

bmRec	bRe	wValue	wInd	wLeng	Data Fragment
0x40	12	0x0088	0 …	1	ff
0x40	12	0x0088	0 …	1	aa
0x40	12	0x0088	0 …	1	55
0x40	12	0x0088	0 …	1	00
0x40	12	0x0088	0 …	1	ff
0x40	12	0x0088	0 …	1	87
0x40	12	0x0088	0 …	1	78
0x40	12	0x0088	0 …	1	e0
0x40	12	0x0087	0 …	1	05
0x40	12	0x0087	0 …	1	04
0x40	12	0x0088	0 …	1	ff
0x40	12	0x0085	0 …	1	dd
0x40	12	0x0085	0 …	1	00
0x40	12	0x0085	0 …	1	00
0x40	12	0x0085	0 …	1	00
0x40	12	0x0085	0 …	1	12
0x40	12	0x0085	0 …	1	00
0xc0	12	0x0084	0 …	1	
0x40	4	0x0082	0 …	8	0000000012000000
0xc0	12	0x0084	0 …	1	
0xc0	12	0x0084	0 …	1	
0x40	12	0x0088	0 …	1	ff
0x40	12	0x0088	0 …	1	aa
0x40	12	0x0088	0 …	1	55
0x40	12	0x0088	0 …	1	00
0x40	12	0x0088	0 …	1	ff

Zeitdiagramm



- an bestimmten Stellen auf Bereitschaft des Geräts warten, sonst Absturz

Funktion der jeweiligen Bytes

- Wie kann man ohne Dokumentation herausfinden, wie z.B. die Scanauflösung eingestellt werden kann?
- Zweimal scannen, nur diesen Parameter variieren
- Beispiel Auflösung, Abschnitt der übertragenen Bytefolge:
  - 300 dpi: `000f2c01802004000100000001801000`
  - 1200 dpi: `000fb004802004000100000001801000`
- nur wenige Parameter müssen verstanden werden, um das Gerät nutzbar zu machen

Bildverarbeitung

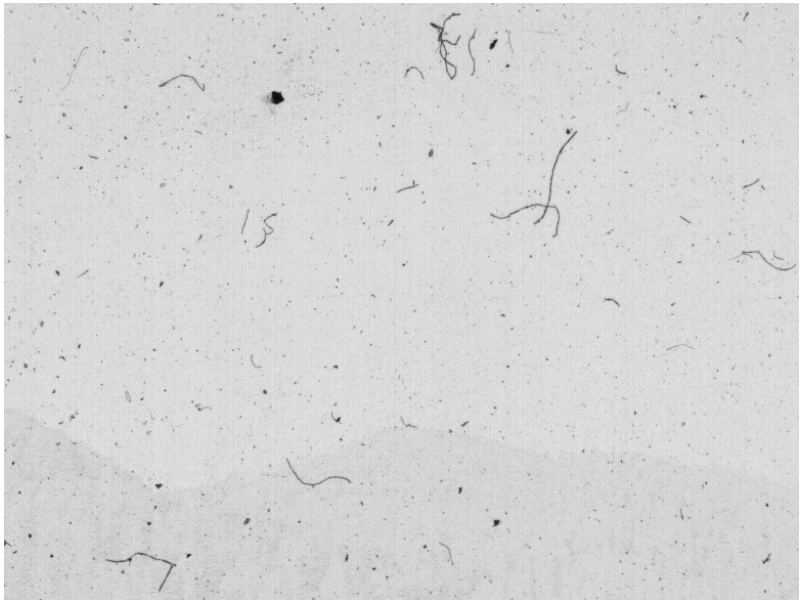
- Helligkeitsunterschiede pro Spalte ausgleichen
  - Helligkeit pro Spalte unterschiedlich
  - Scanner nimmt Leerbild auf
  - Ausgleichen: Division, nicht Subtraktion!

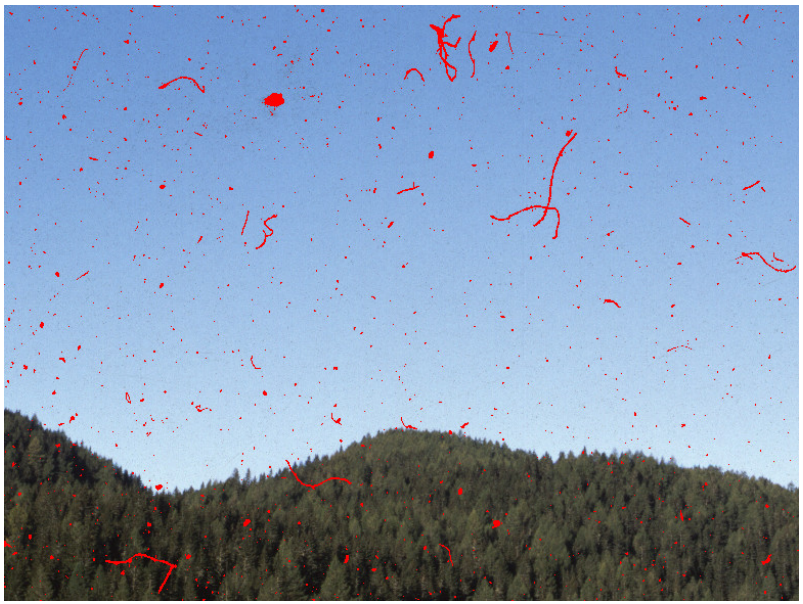

/

=


- Gammakorrektur, Negativmaske, Wertebereich
- automatisches Zuschneiden

Staub-/Kratzerkorrektur





Ausblick: SANE

- Linux-Scannerframework mit GUI und Client/Server
- hat Treiber für CrystalScan 7200, funktioniert aber nicht bei diesem Exemplar
- zumindest Grundfunktionen (Scannen mit einstellbarer Auflösung) in SANE nutzbar machen

Live-Demo