Die Entwicklung eines Wireshark-Dissectors für das BAOS-Kommunikationsprotokoll

Projekt durchgeführt von: Adam Rigely

Betrieb: [FIRMENNAME]

Gliederung der Präsentation

- Vorstellung des Betriebs
- KNX & BAOS – Kontext
- Motivation für das Projekt
- Projektplanung
- Entwurfsplanung
- Implementierung
- Testing
- Soll / Ist - Vergleich
- Fazit

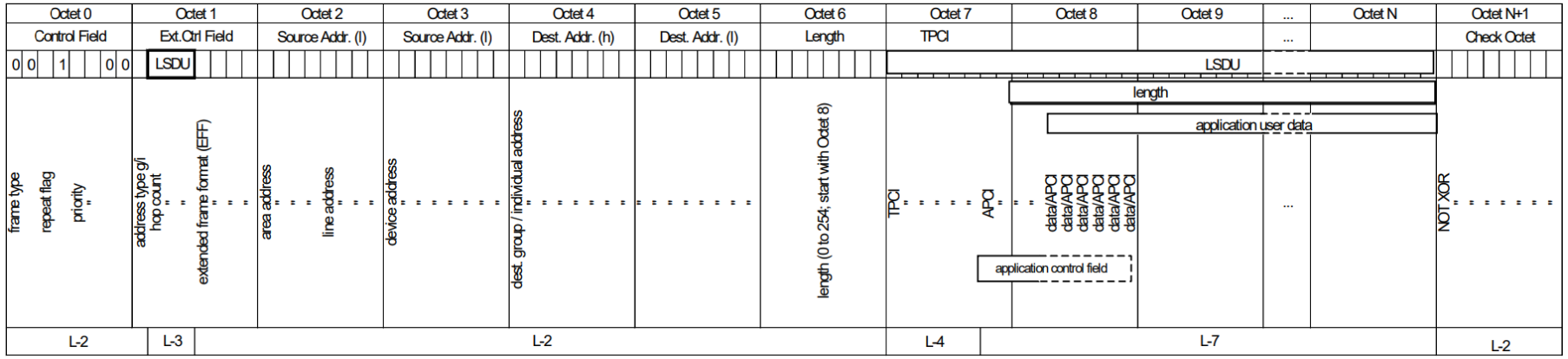
Vorstellung der [FIRMENNAME]

- Anbieter von Automationssystemen in der Gebäudetechnik
- Alle Aspekte der Gebäudeautomation angeboten
- Zonenregler unterstützen unterschiedliche Kommunikationsprotokolle (BACnet, Modbus)

[BILD VON FIRMA]

Unterstützung des KNX-Protokolls

- Das KNX-Protokoll ist kompliziert
- BAOS-Protokoll von Weinzierl Engineering GmbH
- Abstrahierung des KNX-Protokolls
- Kommunikation mit KNX-Geräten einfacher



KNX Extended Frame Format – KNX Specification v2.1, KNX Association

Motivation für das Projekt

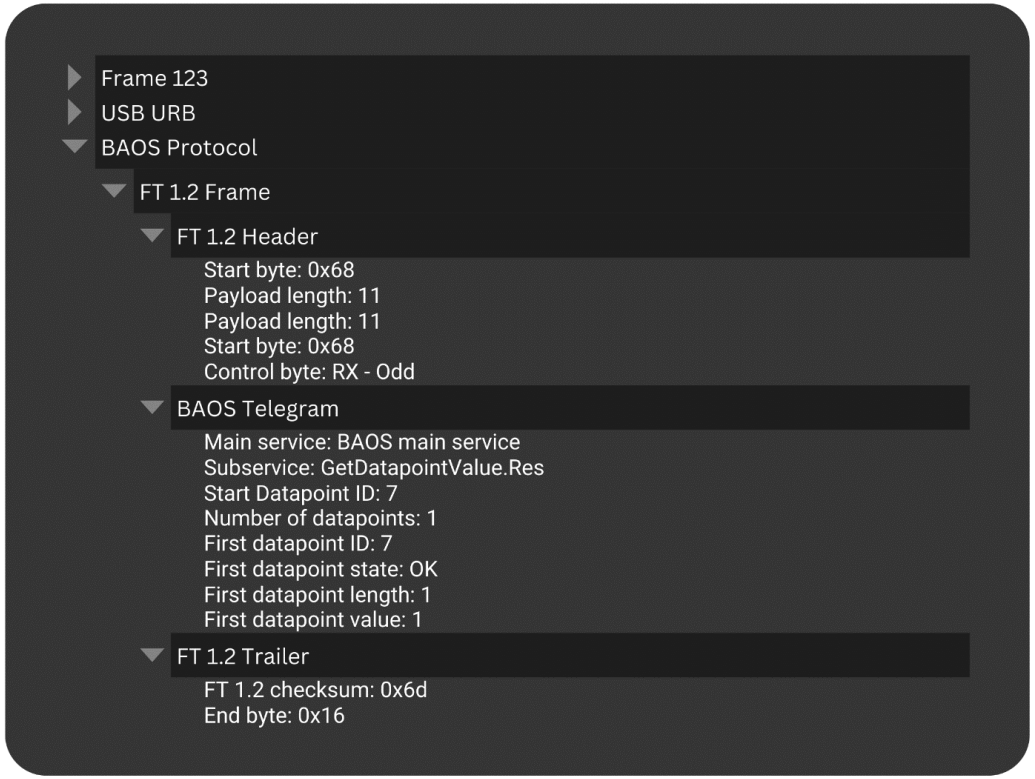
- Entwicklung einer hauseigenen Implementierung des BAOS-Protokolls
- Regelmäßiges Testen nötig
- Netzwerkanalyse-Tool Wireshark
- Manuelle Zerlegung aufwendig
- Keine BAOS-Unterstützung

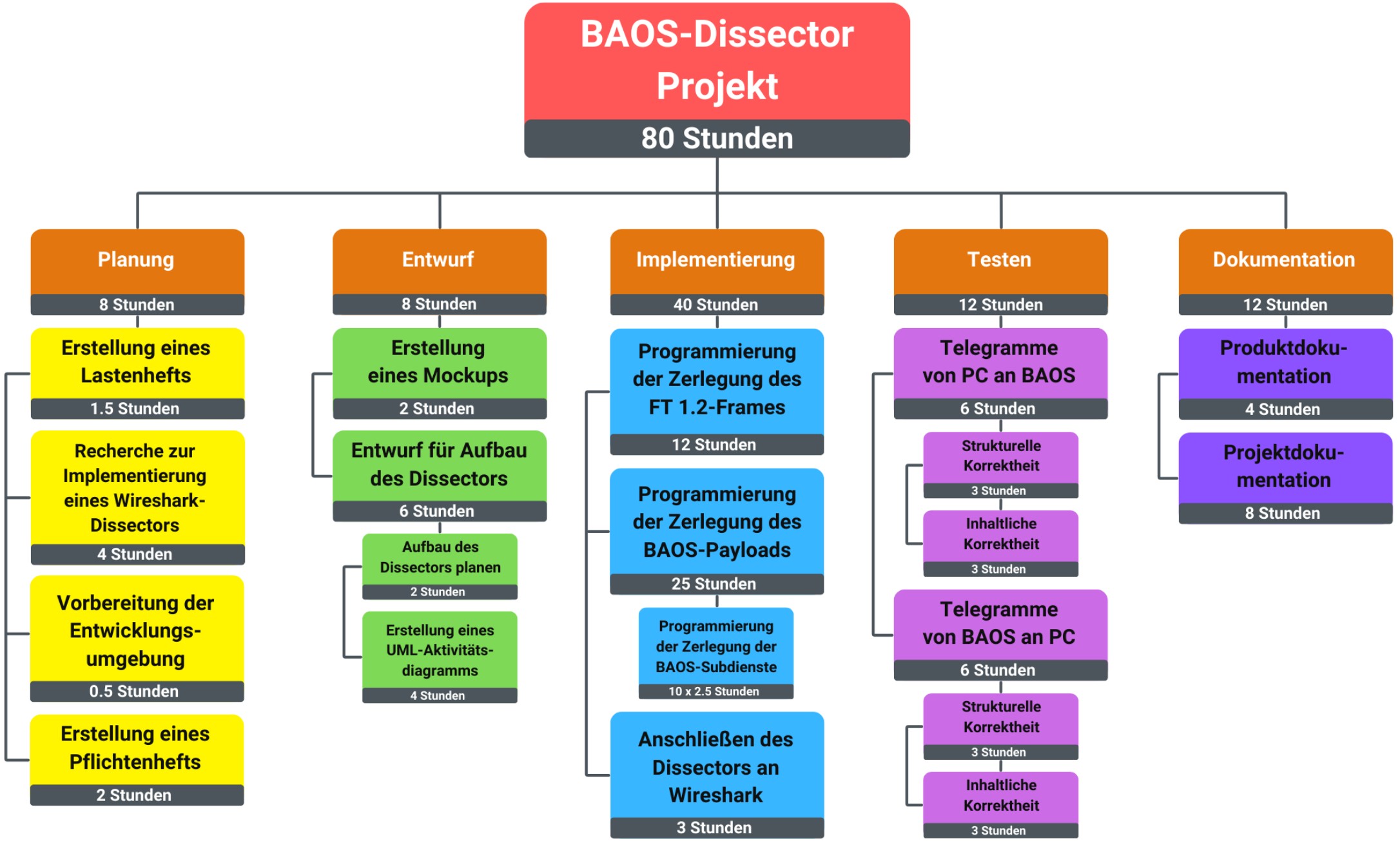
```
1b 00 70 82 4e 31 88 e2  ff ff 00 00 00 00 09 00
01 01 00 02 00 81 03 13  00 00 00 01 60 68 0b 0b
68 d3 f0 81 00 07 00 01  00 07 01 10 64 16
```

Wireshark Dissector

- Zerlegung und strukturierte Darstellung der Pakete
- Optimale Übersicht über Inhalt

```
1b 00 70 82 4e 31 88 e2  ff ff 00 00 00 00 09 00
01 01 00 02 00 81 03 13  00 00 00 01 60 68 0b 0b
68 d3 f0 81 00 07 00 01  00 07 01 10 64 16
```




BAOS-Dissector Projekt
80 Stunden
Planung
8 Stunden
Erstellung eines Lastenhefts
1.5 Stunden
Recherche zur Implementierung eines Wireshark-Dissectors
4 Stunden
Vorbereitung der Entwicklungs-umgebung
0.5 Stunden
Erstellung eines Pflichtenhefts
2 Stunden
Entwurf
8 Stunden
Erstellung eines Mockups
2 Stunden
Entwurf für Aufbau des Dissectors
6 Stunden
Aufbau des Dissectors planen
2 Stunden
Erstellung eines UML-Aktivitäts-diagramms
4 Stunden
Implementierung
40 Stunden
Programmierung der Zerlegung des FT 1.2-Frames
12 Stunden
Programmierung der Zerlegung des BAOS-Payloads
25 Stunden
Programmierung der Zerlegung der BAOS-Subdienste
10 x 2.5 Stunden
Anschließen des Dissectors an Wireshark
3 Stunden
Testen
12 Stunden
Telegramme von PC an BAOS
6 Stunden
Strukturelle Korrektheit
3 Stunden
Inhaltliche Korrektheit
3 Stunden
Telegramme von BAOS an PC
6 Stunden
Strukturelle Korrektheit
3 Stunden
Inhaltliche Korrektheit
3 Stunden
Dokumentation
12 Stunden
Produktdoku-mentation
4 Stunden
Projektdoku-mentation
8 Stunden

Kostenanalyse

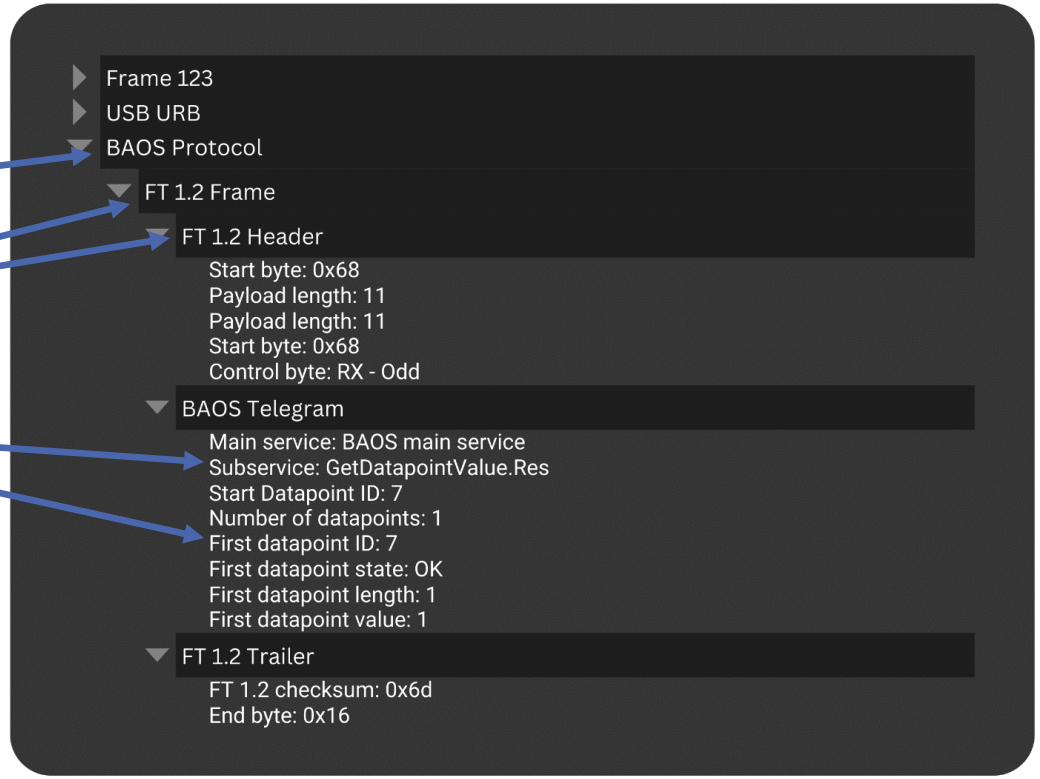
	Zeitdauer	Kosten	Kosten gesamt
Stundensatz Praktikant	80 Stunden	22,15 € / Stunde	1.772,00 €
Stundensatz Ausbilder	2 Stunden	22,15 € / Stunde	44,30 €
Canva Pro Abonnement	1 Monat	12,00 € / Monat	12,00 €
Pauschalbetrag Weitere Ressourcen (Strom, Hardware Abnutzung etc...)	80 Std / 4 Std = 20 Tage	5 € / Tag	100,00 €
1.928,30 €			

Amortisationsdauer

	Vor Einsatz Dissectors	Nach Einsatz Dissectors	Ersparnis
Analyse eines Paketes	≈ 2 Min	≈ 0,5 Min	1,5 Min
Paketanalyse im Monat	≈ 10 Std	≈ 2,5 Std	7,5 Std
Kosten der Paketanalyse im Monat	≈ 221,50€	≈ 55,38€	166,12€
Amortisationsdauer	1928,30€ / 166,12€ = 11,61 Monaten ≈ 1 Jahr		

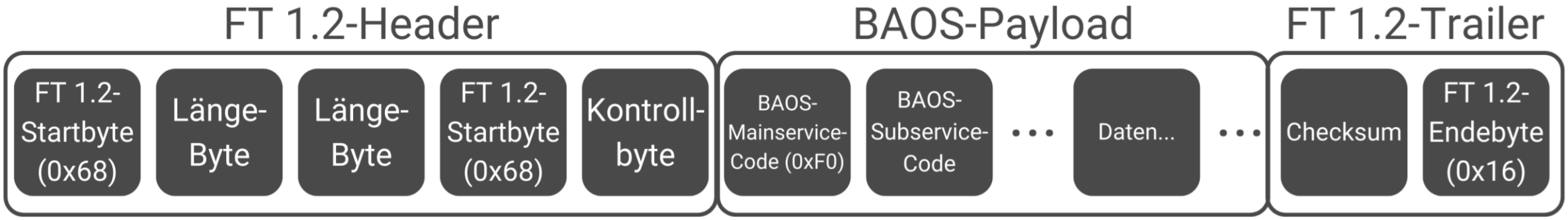
Recherche nach Dissectoren

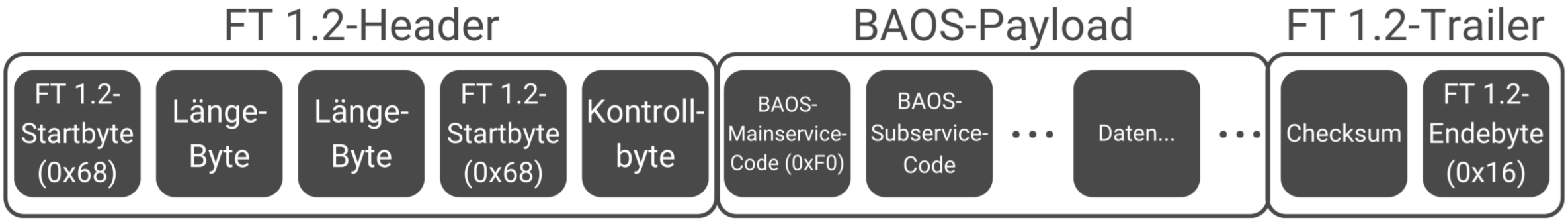
- Unterstützte Sprachen: C, Lua
- Aufbau eines Dissectors
- Registrierung des Protokolls
- TreeItems (Baum-Objekte)
- ProtoFields (Protokollfelder)



Registrierung des Protokolls

- Registrierung anhand Portnummer
  (HTTPS – 443, SSH – 22, BACnet – 47808)
- BAOS über serielle Schnittstelle
- Lösung: heuristischer Dissector
- Registrierung anhand Patterns

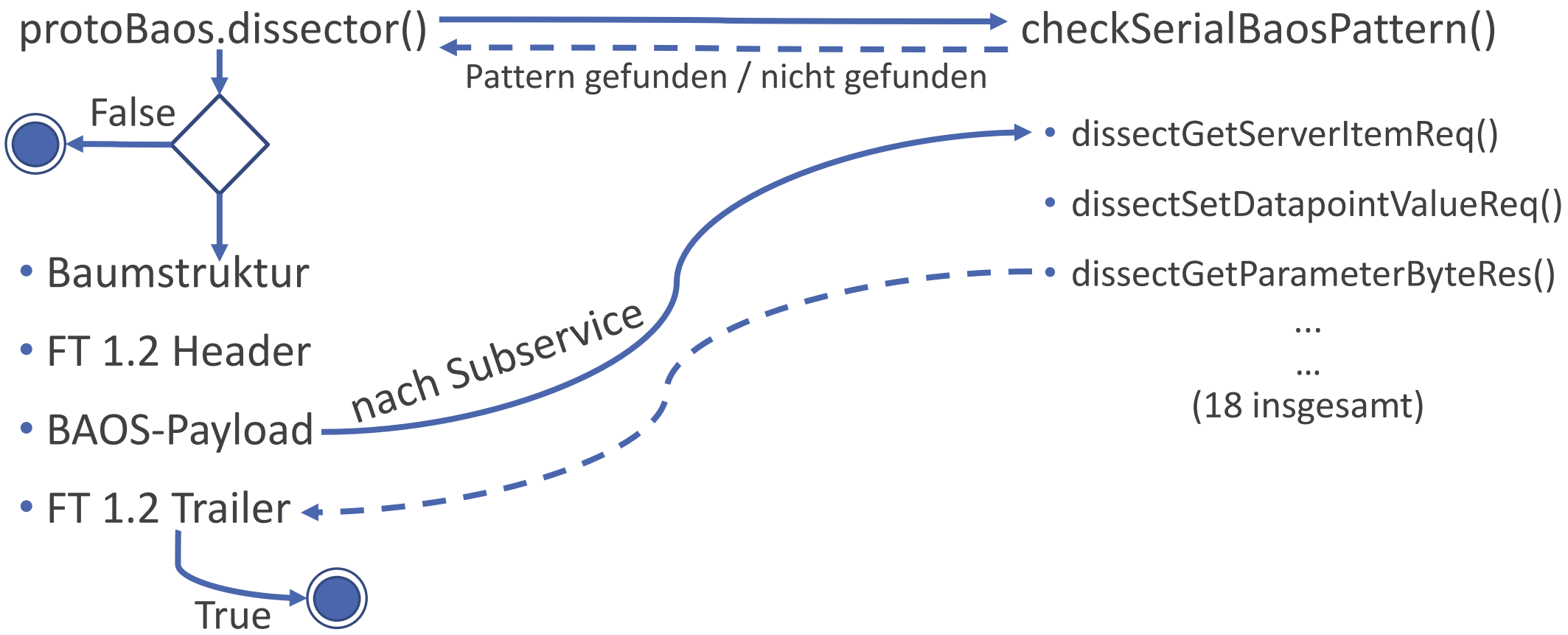




1) Startbyte gefunden?
2) Zweites Startbyte vorhanden?
3) Länge-Bytes sind gleich?
4) Kontrollbyte hat einen der vordefinierten Werte?
5) BAOS-Mainservice-Code gefunden?

- 0x73
- 0x53
- 0xF3
- 0xD3

Implementierung



Testing

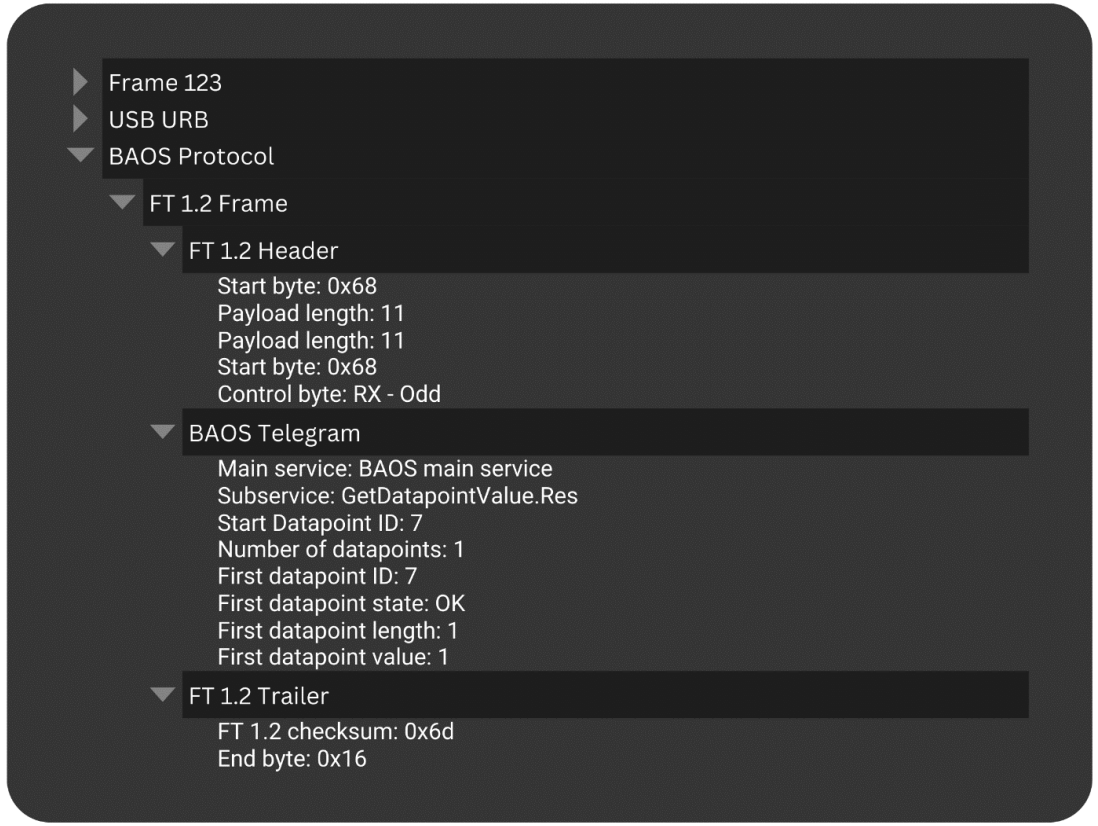
- Während und nach Entwicklung
- Manuelle Tests
- Testfunktionen im BAOS-Protokoll
- Zentrale Funktion

```
void BaosTester::startTestingProcess(
    bool runGetServerItemTest,
    bool runSetServerItemTest,
    bool runGetDatapointDescriptionTest,
                    ...
void BaosTester::testGetServerItem()
{
    printf("TESTING GET SERVER ITEM STARTED\n\n");

    GetServerItem gsi(serialConnection);

    BaosHardwareType baosHardwareType = gsi.getHardwareType();
    printf("Hardware Type single request: %02X %02X %02X %02X %02X %02X\n",
        baosHardwareType.byte1,
        baosHardwareType.byte2,
        baosHardwareType.byte3,
        baosHardwareType.byte4,
        baosHardwareType.byte5,
        baosHardwareType.byte6);
    BaosVersion hardwareVersion = gsi.getHardwareVersion();
    printf("Hardware Version single request: %d.%d\n",
        hardwareVersion.mainVersion,
        hardwareVersion.minorVersion);
                    ...
```

Soll / Ist - Vergleich




```
BAOS
  FT 1.2 frame
    FT 1.2 frame header
        Start byte: 0x68
        Payload length: 13
        Payload length: 13
        Start byte: 0x68
        Control byte: RX - Odd (0xf3)
    BAOS telegram
        BAOS main service: 0xf0
        BAOS subservice: GetDatapointValue.Res (0x85)
        Start Datapoint ID: 9
        Number of datapoints: 1
        Datapoint ID: 9
      0001 0000 = Datapoint state flags: 0x10
          .... ..00 = Datapoint transmit priority: Idle/OK (0x0)
          .... .0.. = Datapoint read request flag: Write request should be sent (0x0)
          .... 0... = Datapoint update flag: Value is not updated (0x0)
          ...1 .... = Datapoint valid flag: Object has already been received (0x1)
        Datapoint length: 2
        Datapoint value: ff f9
    FT 1.2 frame trailer
        FT 1.2 checksum: 0x85
        End byte: 0x16
```

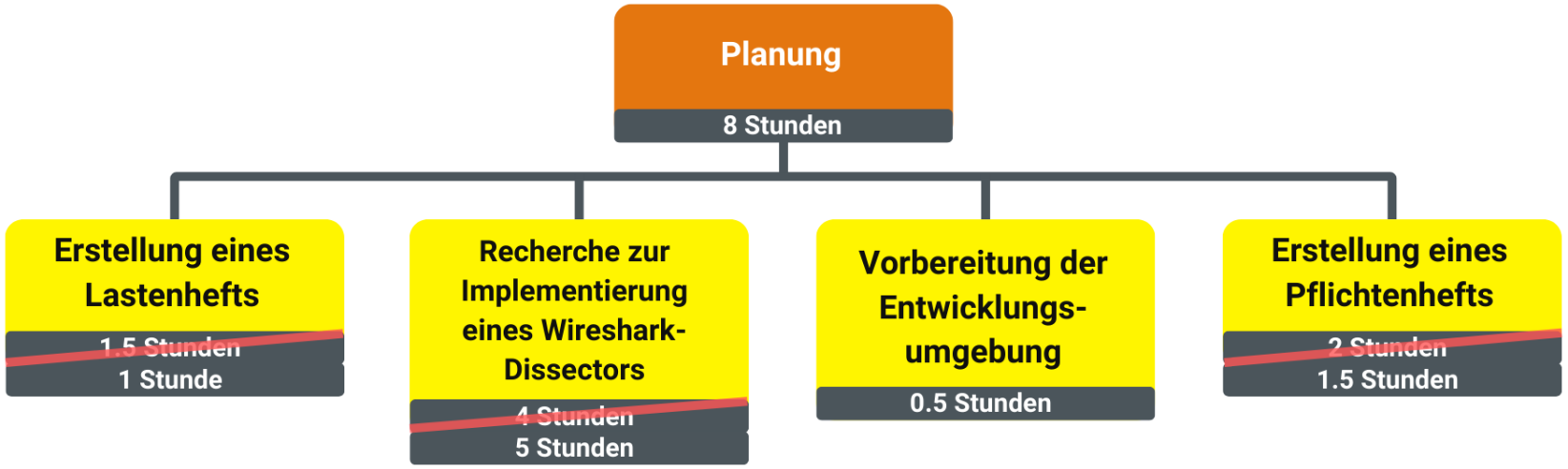
Soll / Ist - Vergleich

- ✓ Protokoll erkannt
- ✓ Baumstruktur
- ✓ Datenpaket korrekt zerlegt
- ✓ Übersichtliche Darstellung

```
BAOS
FT 1.2 frame
  FT 1.2 frame header
    Start byte: 0x68
    Payload length: 13
    Payload length: 13
    Start byte: 0x68
    Control byte: RX - Odd (0xf3)
  BAOS telegram
    BAOS main service: 0xf0
    BAOS subservice: GetDatapointValue.Res (0x85)
    Start Datapoint ID: 9
    Number of datapoints: 1
    Datapoint ID: 9
    0001 0000 = Datapoint state flags: 0x10
      .... ..00 = Datapoint transmit priority: Idle/OK (0x0)
      .... .0.. = Datapoint read request flag: Write request should be sent (0x0)
      .... 0... = Datapoint update flag: Value is not updated (0x0)
      ...1 .... = Datapoint valid flag: Object has already been received (0x1)
    Datapoint length: 2
    Datapoint value: ff f9
  FT 1.2 frame trailer
    FT 1.2 checksum: 0x85
    End byte: 0x16
```

Fazit

- Abgabe
- Endergebnis zufriedenstellend
- Dokumentationen geschrieben



Wie wird der Dissector genutzt

Dies ist eine kurze Präsentation, wie der Dissector genutzt werden kann.

BAOS-Telegramme und -Felder erkennen und beschriften

Während der Überwachung einer Verbindung oder wenn eine abgespeicherte Mitschnittdatei geöffnet ist, wenn Wireshark, mit dem aktivierten BAOS-Dissector, ein BAOS-Telegramm erkennt, dieses Telegramm wird jetzt ordentlich als Paketbaum auf dem *Paketinformationen*-Panel dargestellt. Der FT 1.2-Frame-Header und der -Frame-Trailer um den BAOS-Payload herum werden auch zerlegt.

```
▸ Frame 838: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface \\.\USBPcap1, id 0
▸ USB URB
▾ BAOS Protocol
  ▾ FT 1.2 frame
    ▾ FT 1.2 frame header
        Start byte: 0x68
        Payload length: 11
        Payload length: 11
        Start byte: 0x68
        Control byte: TX - Odd (0x73)
    ▾ BAOS telegram
        BAOS main service: 0xf0
        BAOS subservice: SetDatapointValue.Req (0x06)
        Start Datapoint ID: 1
        Number of datapoints: 1
        Datapoint ID: 1
        Datapoint command byte: Set new value and send on bus (0x03)
        Datapoint length: 1
        Datapoint value: 01
    ▾ FT 1.2 frame trailer
        FT 1.2 checksum: 0x71
        End byte: 0x16
```

Der BAOS-Dissector integriert sich ordentlich in Wireshark. Das heißt, alle dazugehörigen Funktionalitäten werden unterstützt.

Wenn Protokollfelder auf dem *Paketinformationen*-Panel selektiert werden,

```
▾ BAOS telegram
    BAOS main service: 0xf0
    BAOS subservice: SetDatapointValue.Req (0x06)
    Start Datapoint ID: 1
    Number of datapoints: 1
    Datapoint ID: 1
```

die dazugehörigen Hexadezimalwerte werden ebenso hervorgehoben,

```
f0 06 00 01 00 01 00 01
```

und der Feldname, der Filtername, die Wertgröße werden auf der Statusleiste angezeigt.

```
Number of datapoints (baos.nrOfDps), 2 bytes
```

Ausblick

- C-Version
  - Leistung
  - Konfigurierbarkeit

Literaturverzeichnis

- KNX Specification v2.1, KNX Association
- [FIRMENWEBSITE]
- https://weinzierl.de/

Vielen Dank für Ihre Aufmerksamkeit!

Q & A