Entwicklung einer plattformunabhängigen und ressourcenschonenden Anbindung von Lasermesssystemen

von

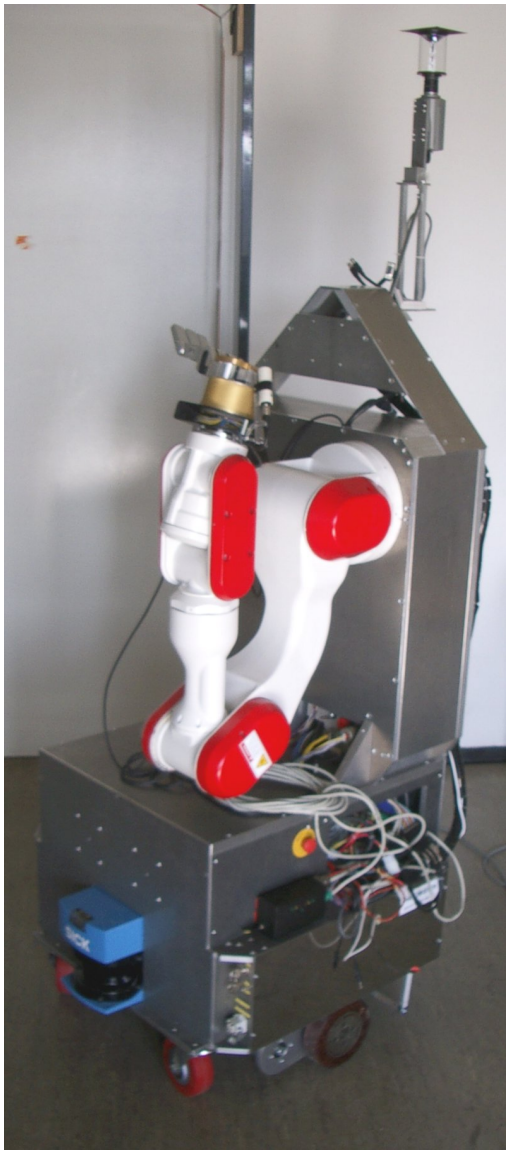
Hannes Bistry und Stephan Pöhlsen

Überblick

- Sick LMS 200
- Bisherige Anbindung
- Anforderungen
- Rabbit PowerCore 3800
- Softwaredesign
  - Vorverarbeitung
- Benchmark
- Ausblick

Service-Roboter

- Omnivision Kamera
- Stereovision Kamerasystem (PTU)
- Mitsubishi PA10
- BarrettHand
- Handkamera
- Differentialantrieb
- zwei Sick LMS 200



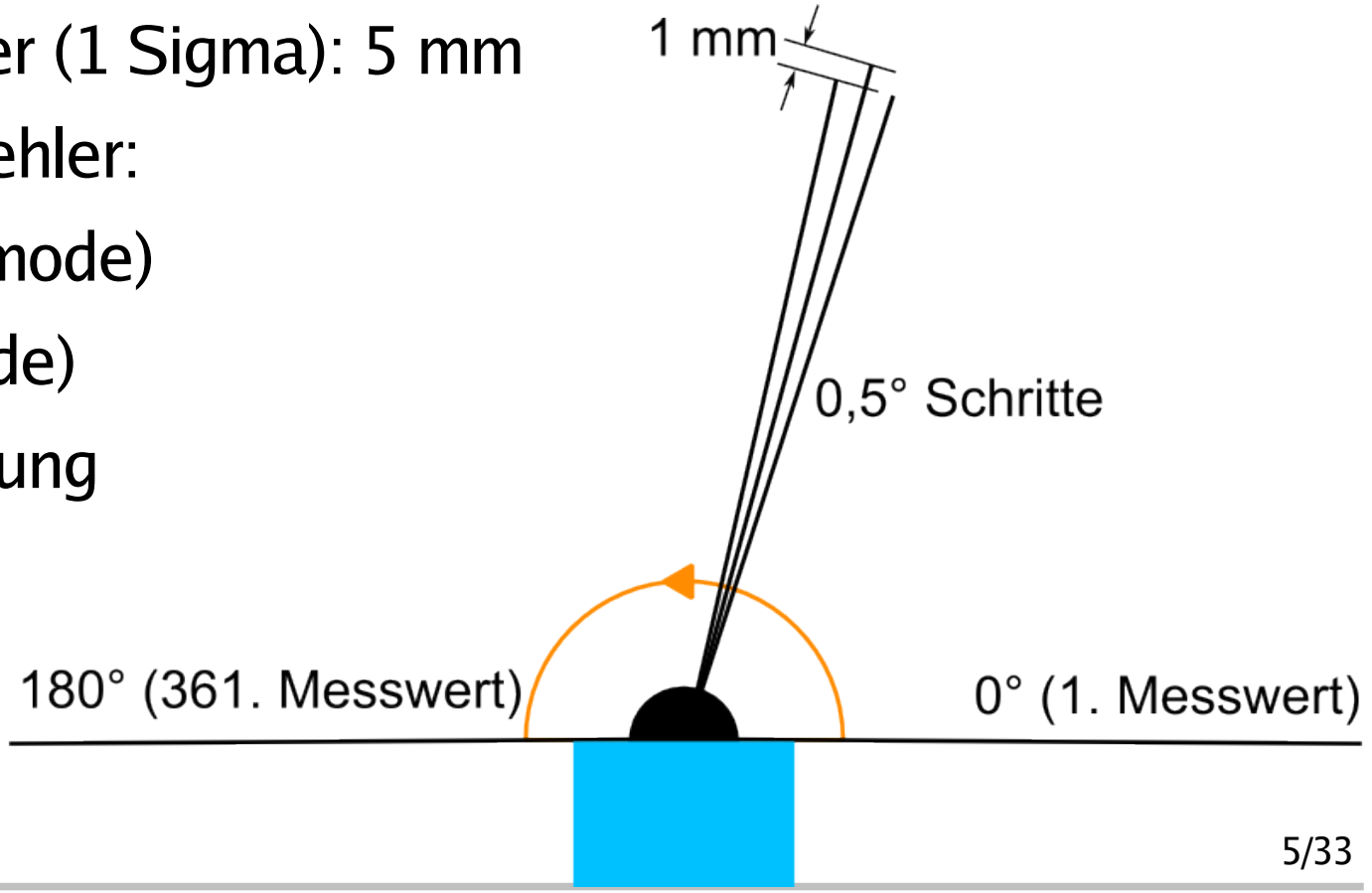
Sick LMS 200

- Einsatzgebiete
  - Volumen- / Konturbestimmung
  - Flächenüberwachung (Objektschutz)
  - Kollisionsschutz
  - Positionsermittlung

- Merkmale und Vorteile
  - berührungslose optische Messung
  - kurze Tastzeit
  - keine Remissionseigenschaften erforderlich
    - beliebiger Hintergrund
    - Marken optional
  - keine Ausleuchtung notwendig
  - Messdaten in Echtzeit

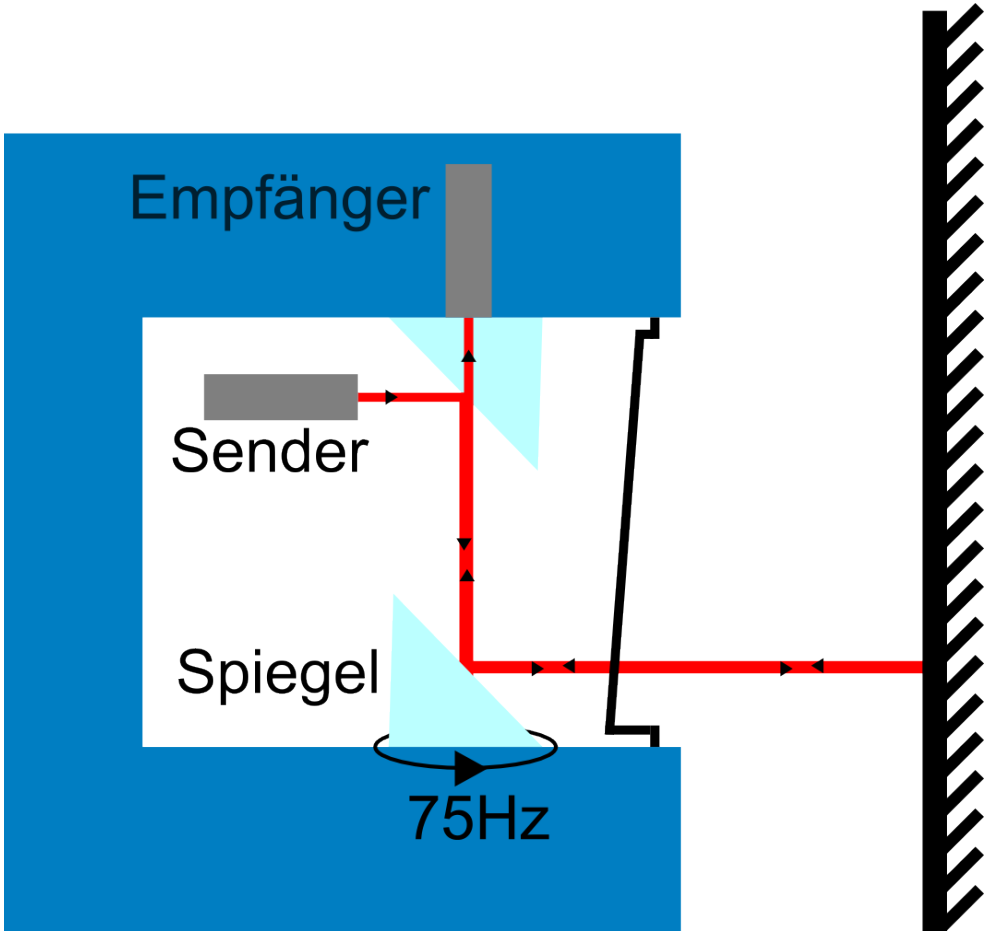
Messauflösung

- Auflösung: 10 mm oder 1 mm
- statistischer Fehler (1 Sigma): 5 mm
- systematischer Fehler:
  - ±20 mm (mm-mode)
  - ±4 cm (cm-mode)
- 0,5° Winkelauflösung



Time of Flight

- Laserimpuls aussenden
  - 905 nm (Infrarotbereich)
- Zeitmessung bis zum Eintreffen des Echos
- $$Abstand = \frac{Zeit * Lichtgeschwindigkeit}{2}$$
- zusätzlich Speicherung der Reflexionsstärke möglich



Sick LMS 200 Schnittstelle

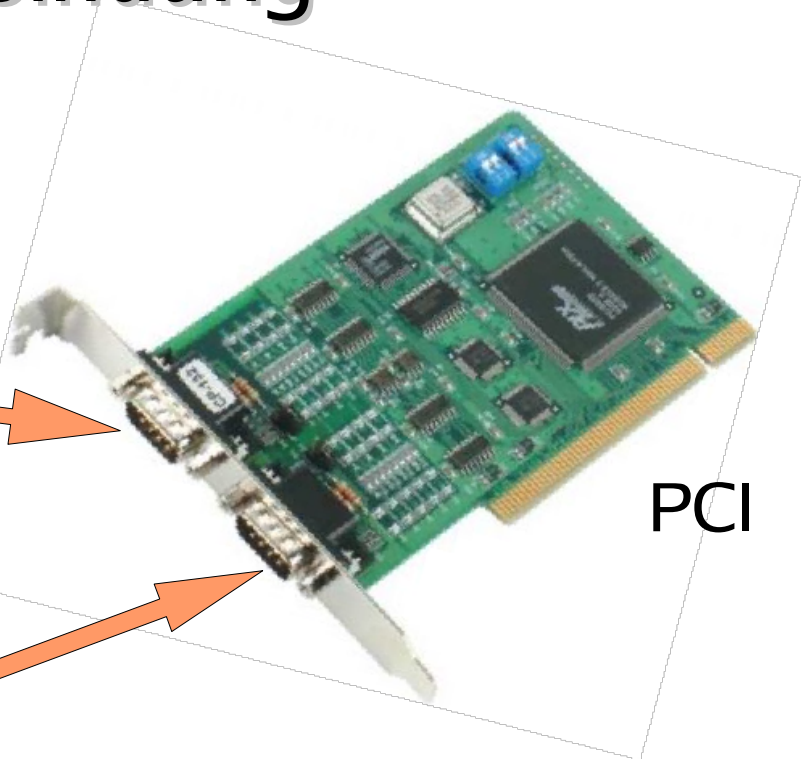
- Hardware: RS-232 / RS-422
  - mit Jumper im Stecker einstellbar
  - Echtzeitauswertung nur mit RS-422 (500k Baud)
    - RS-232 nur 38,4k Baud
- Software: Telegramme
  - Header: Adresse, Telegrammlänge
  - Daten: Kommando, Subdaten, Status (bei Antworten)
  - CRC-Prüfsumme

Telegramme

- Status
  - request (0x31): fordert vom LMS einen Status an
  - response (0xB1): 142 Bytes Statusinformationen
- Mode
  - request (0x20)
    - 0x24 „scan continuously“
    - 0x25 „scan if requested“
    - 0x40 = 38K4, 0x41= 19K2, 0x42 = 9K6, 0x48 = 500K
  - response (0xA0): 1 Byte (successful)
- Data (0xB0): 732 Byte: mm/cm, Messwerte, Status-Byte

Bisherige Anbindung

RS-422



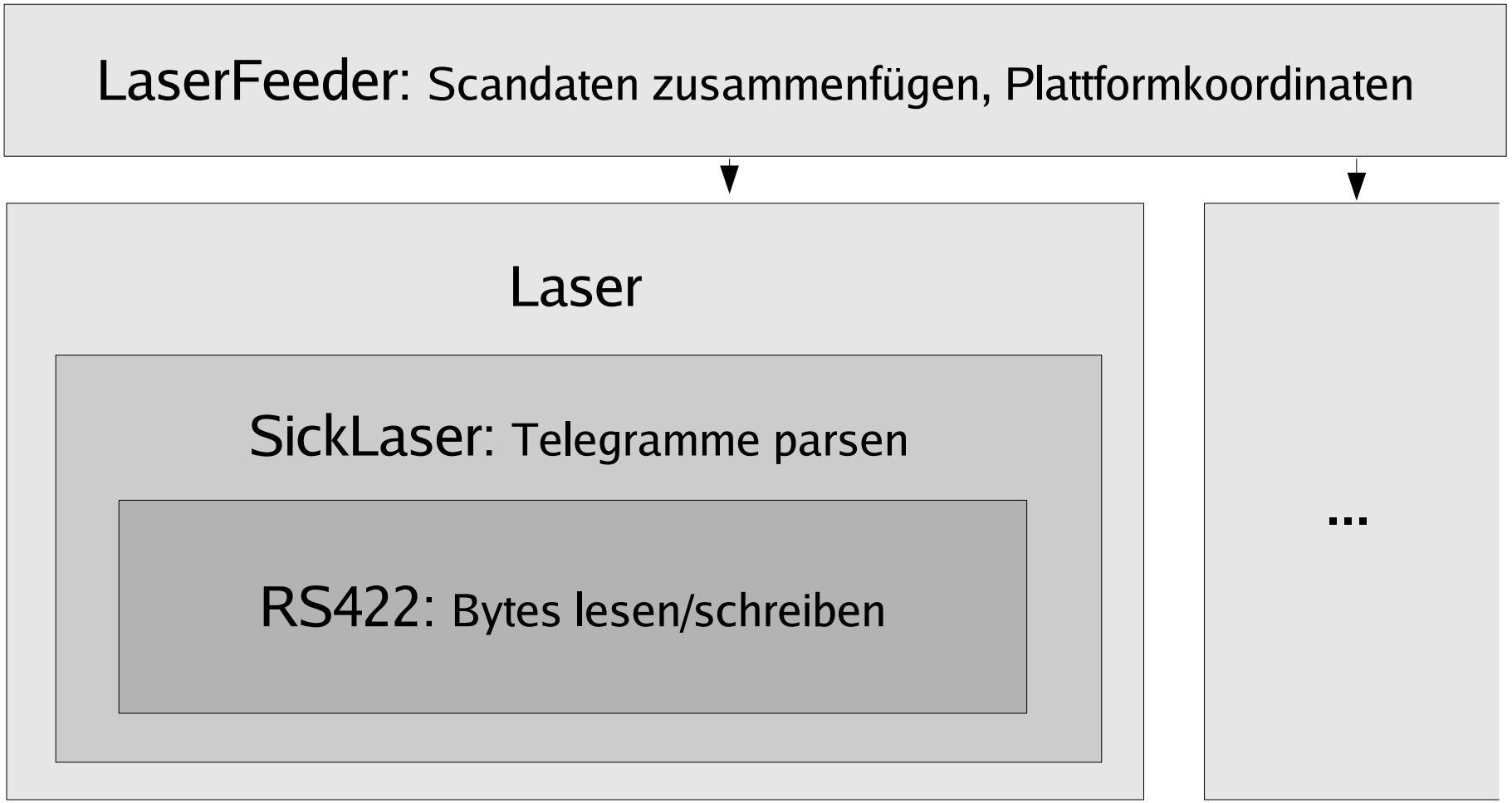
PCI

Moxa
Schnittstellen-
Karte

Softwareschnittstelle

- LaserFeeder Thread
  - GetClosestObstacleDistance()
  - GetLaserScanScanner()
  - GetLaserScanPlatform()
  - GetLaserScanPlatformMatched()
  - GetNumScanners()
  - GetScannerPosition()
- GenBase->UpdateLaser()

Bisherige Implementierung



Probleme der Moxa-Anbindung

- Hohe Systemauslastung (Messzeitraum: 1 Minute)
  - 23,4 % system + 52,3 % user
  - 14,4 % für inb()
- Paketverluste (Messzeitraum: 5 Minuten)
  - Scanner0: 36,2 %
  - Scanner1: 40,8 %
  - Byteverlustrate ca. 0,15 %
- (Plattformabhängig: Treiber, PCI-Slot)

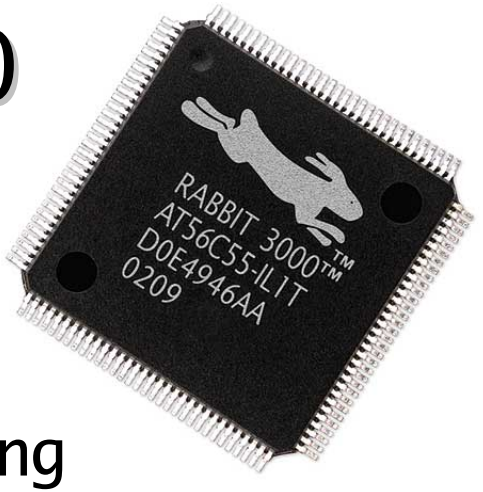
Anforderungen

- zwei serielle 500kBaud Schnittstellen
- RS-422 konforme Pegel
- möglichst plattformübergreifend bezüglich
  - Treiber/Betriebssystem/Architektur
  - Hardwareschnittstelle
- keine Verluste von Messdaten
- geringe Systemlast auf dem PC
- platzsparend
- preiswert

Hardware Lösungsansätze

- Embedded System zwischen Scannern und PC
- Art der Realisierung
  - FPGA (hohe Parallelisierbarkeit, Vorverarbeitung)
  - Mikrocontroller (günstig, einfach programmierbar)
- Schnittstelle zum Rechner:
  - Ethernet (an PCs vorhanden)
  - USB (auf vielen Entwicklerboards vorhanden)

Rabbit PowerCore 3800



- Ethernet
- Rabbit 3000 CPU mit 51,6 MHz Takt
- 6 serielle Schnittstellen mit x8 Überabtastung
  - (51,6 MHz / 8 / 13 = 496 KHz)
- 169,- EUR (Entwicklungskit)

Spezifikationen des Rabbit 3000

- 8-bit Mikroprozessor mit einigen 16-bit Operatoren
- Weiterentwicklung des Zilog Z80, ähnliche Architektur, hohe Abwärtskompatibilität zum Z80-Code
- Leistungsaufnahme ca. 0,34 Watt bei 51,6 MHz
- Timer
  - Event-Steuerung
  - Taktung der seriellen Schnittstellen
- Interrupts auf 3 verschiedene Prioritätsstufen
  - ISR höherer Priorität können die niedriger Priorität abbrechen
    - Effizientes Design von Echtzeitanwendungen

Spezifikation des Rabbit 3000 (2)

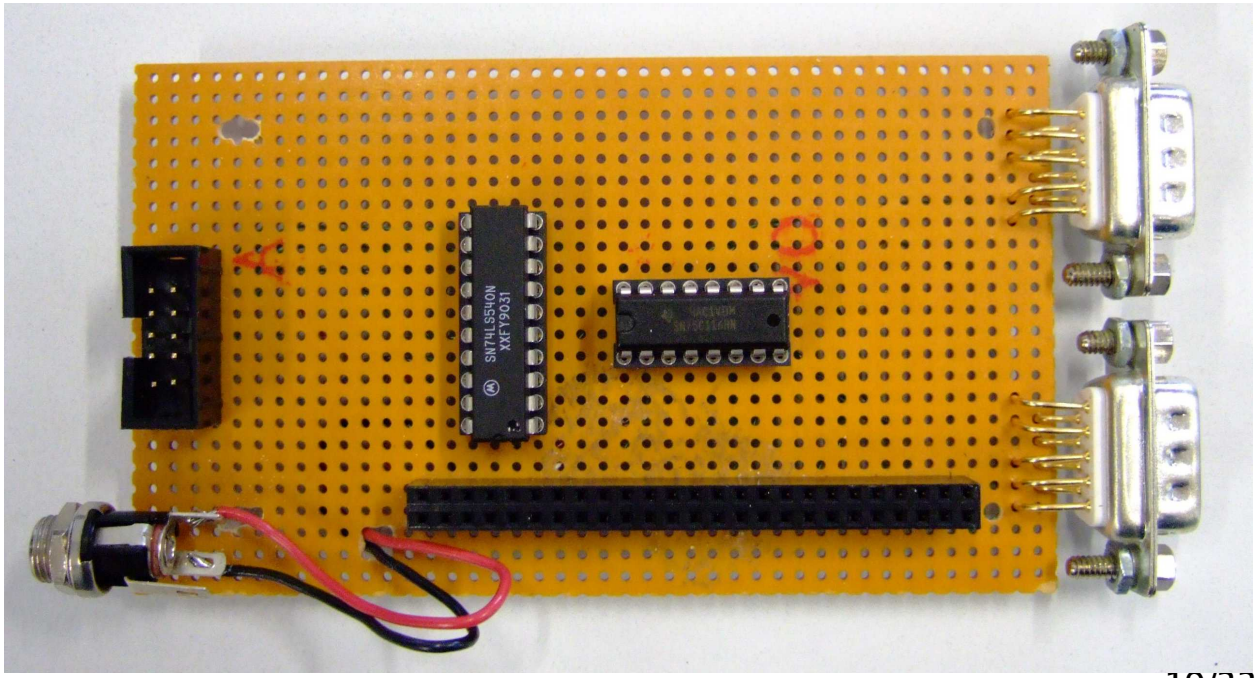
- 6 serielle Schnittstellen (IrDA fähig)
  - Asynchrone Datenübertragung bei allen möglich mit Datenraten bis zu (Prozessortakt/8)
  - 4 davon unterstützen synchrone Datenübertragung
  - 2 mit 4 Byte FIFO
- 7 parallele Schnittstellen (8-bit)
- 2 Input Capture Units (Zeitmessung zwischen 2 Events)
- 2 Encoder Inputs (mit Drehrichtungserkennung)
- Die Ausgangspins des Rabbit sind überbelegt->
  - nicht alle Funktionen gleichzeitig nutzbar

PowerCore 3800

- Speicher:
  - 512 kB Flash
  - 1 MB SRAM (512 kB Code, 512 kB Data)
  - 1 MB Serial-Flash
- Ethernet Controller
- Spannungswandler (8-40 V DC, 24 V AC)
- Leistungsaufnahme ca. 1,5 W (incl. Rabbit)

Serielle Schnittstelle

- Pegel: Rabbit 3000 hat CMOS Pegel (0 V und 3.3 V)
- Umsetzung auf RS-422 Pegel erfolgt über einen Transceiver Chip (Typ SN75C1168, 2x receiver, 2x transmitter)
- Auf Tochterplatine mit Sub-D Buchsen verlötet, Stromversorgung über PowerCore



Programmierung

- Proprietäre Entwicklungsumgebung Dynamic C für Windows, an eingebettete Systeme angepasst
  - Ähnlich ISO/ANSI C, mit einigen Änderungen
  - Konstrukt der Costatements/Cofunctions
    - Erlauben nebenläufiges Ausführen von Prozessen in einem einzelnen Programm
    - Kontrolle wird dem nächsten Prozess übergeben, sobald ein Wartezustand eingenommen wird
  - Es kann passagenweise direkt in Assembler programmiert werden

Programmierung (2)

- Serielles Programmierkabel
- Booten direkt über die serielle Schnittstelle
  - Programm wird in den Flash geschrieben
  - im RAM ausgeführt
  - Debugging zur Laufzeit
- Ist kein Programmierkabel angeschlossen, bootet der Rabbit direkt aus dem Flash-Speicher
- In-Circuit-Programmable

Softwareanforderungen

- SICK LMS 200
  - unabhängiger Betrieb zweier LMS
  - selbstständige Synchronisation und Konfiguration
  - automatisches Erkennen von Verbindungsproblemen und gegebenenfalls Resynchronisation
- Daten über Ethernet-Schnittstelle weiterleiten
- Kennzeichnung der Daten (Scannerzugehörigkeit)

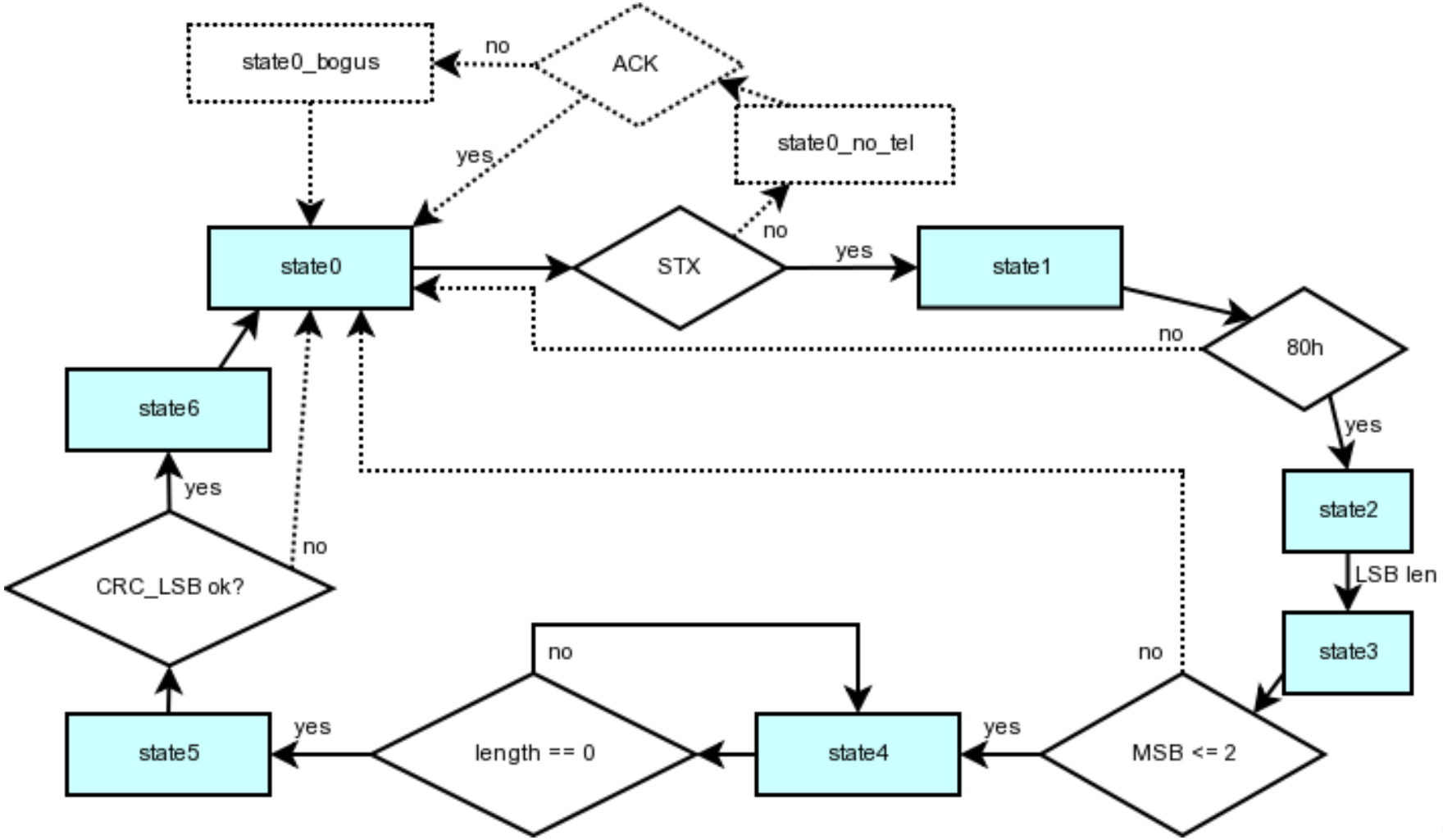
Software-Design

- Telegramme in ISR synchronisieren
  - Zustandsautomat
  - „Zero-Copy“
  - extrahieren einzelner Telegramme ermöglichen
- CRC in ISR
  - Lastglättung durch sukzessives Berechnen der Prüfsumme
  - Bei einem Byteverlust (durch Störungen) kann das nächste Telegramm erkannt werden
- Vorverarbeitung und Weiterleitung außerhalb der ISR (ohne Priorität)

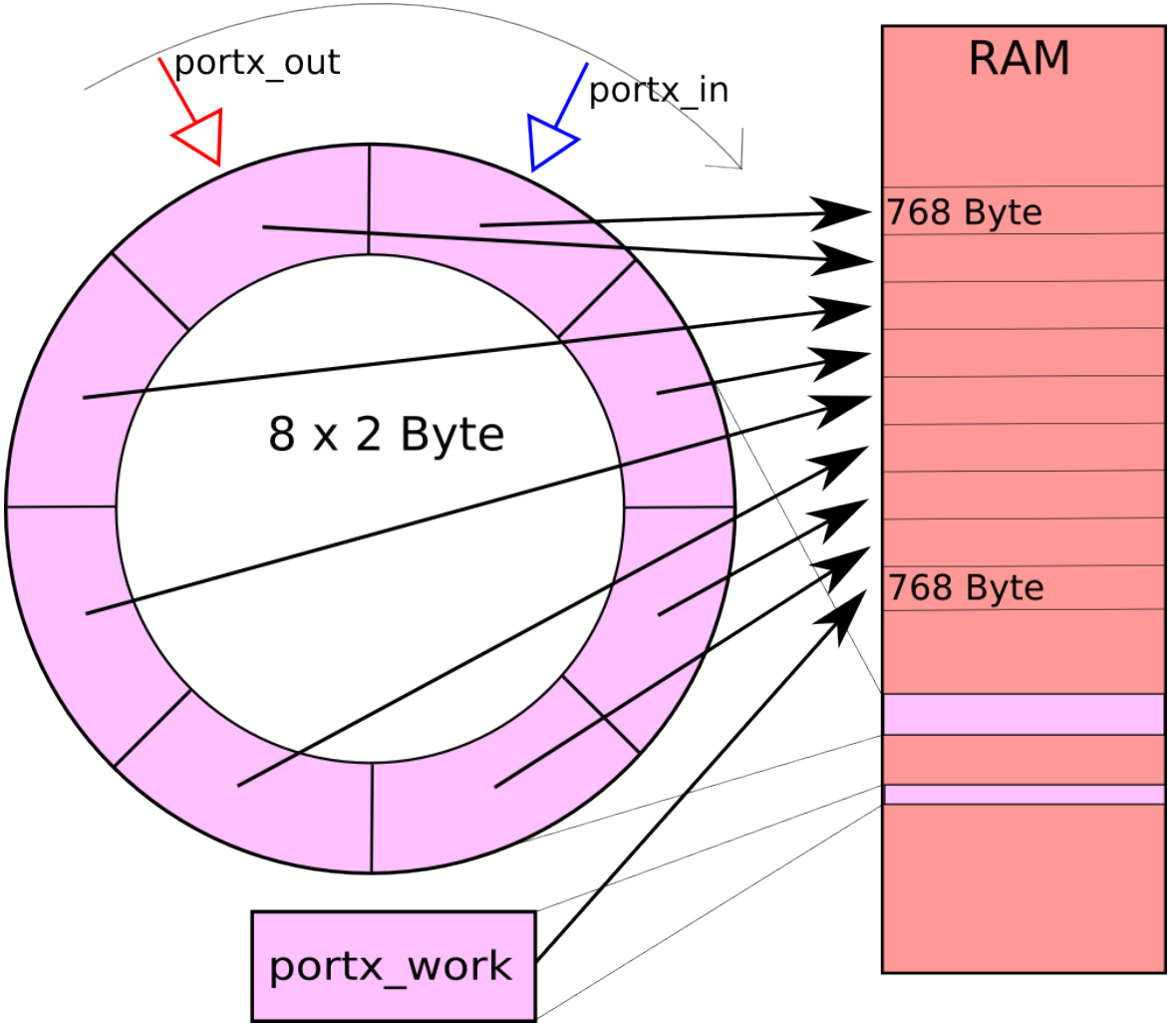
Telegramm <-> ISR-Zustände

<table>
<tr><td>STX</td><td>ADR</td><td colspan="2">LENGTH</td><td>CMD</td><td>DATA</td><td>STATUS</td><td colspan="2">CRC</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">2</td><td>1</td><td>LENGTH-2</td><td>1</td><td colspan="2">2</td></tr>
<tr><td>state0</td><td>state1</td><td>state2</td><td>state3</td><td colspan="3">state4</td><td>state5</td><td>state6</td></tr>
</table>

Telegrammsynchronisation in der ISR



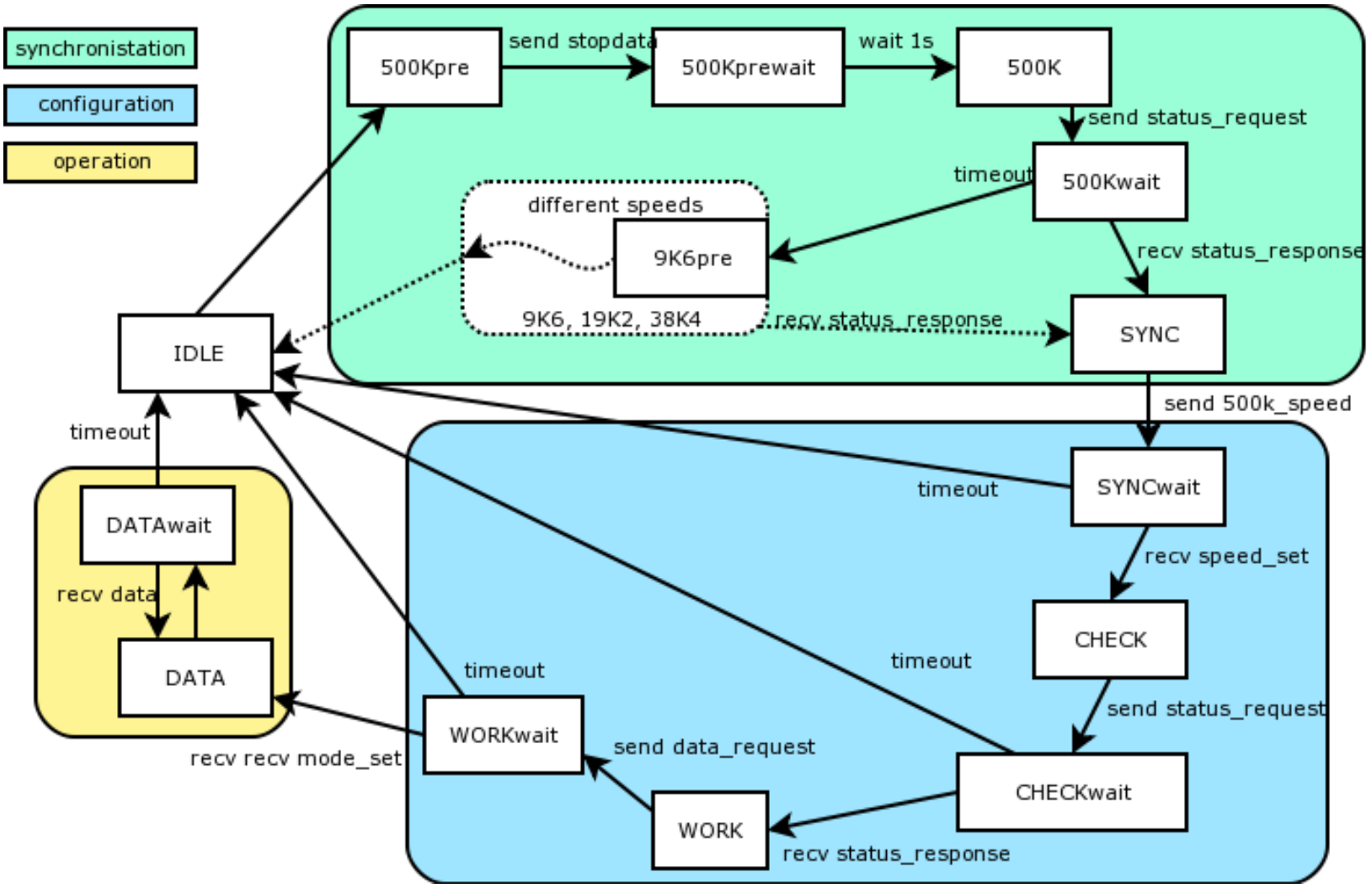
Telegrammpuffer



Empfangssicherheit

- 2x 500 kBd / 8N1 => 2x 50 kByte/s
- pro Byte maximal **279 cycles** in der ISR
- bei 51,6 MHz sind ca. 185 kByte/s an Durchsatz möglich
  - demnach maximal 55 % Systemauslastung
- 732 Byte (Messdatentelegramm) * 37,5 (Telegramme/Sek.)
  - 2x 27,5 kByte/s
  - ca. **30%** Systemauslastung für reinen Datenempfang

Zustandsautomat für die LMS



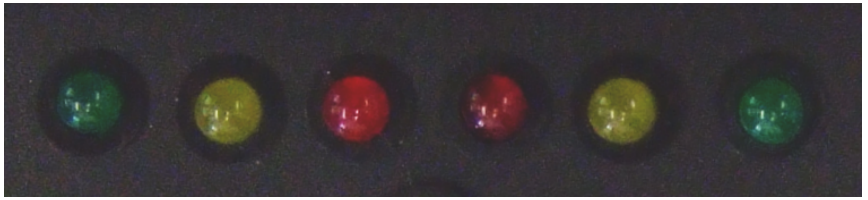
Neue Implementierung der Softwareschnittstelle

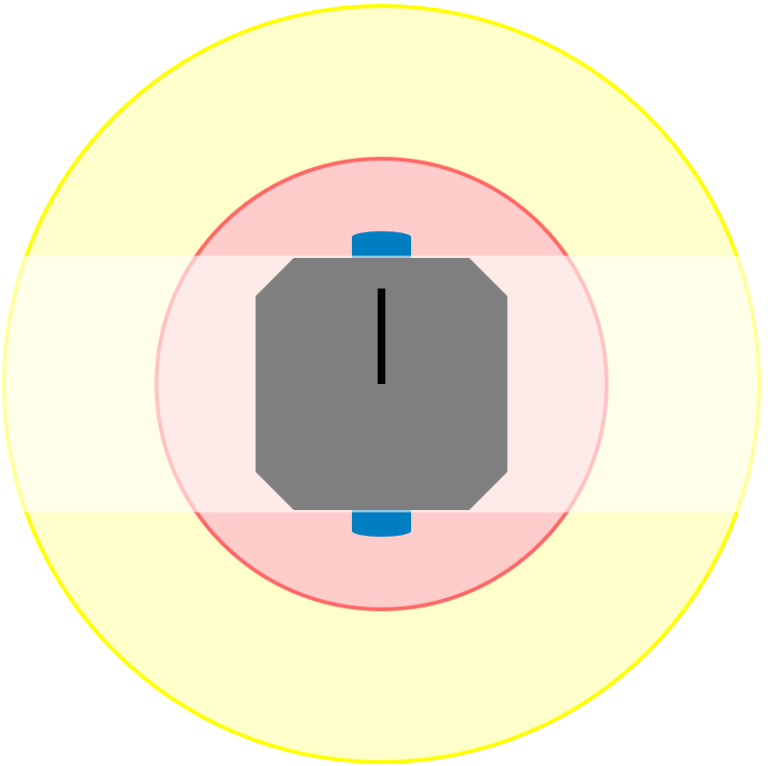
- UDP-Socket binden
- Main-Loop
  - UDP-Paket empfangen
  - in RadialScan Objekt speichern
  - Markenupdate an GenBase senden
- Liefert für die Methoden die gespeicherten Daten zurück

LaserFeeder:
Scandaten zusammenfügen, Plattformkoordinaten

Vorverarbeitung: Kollisionswarnung

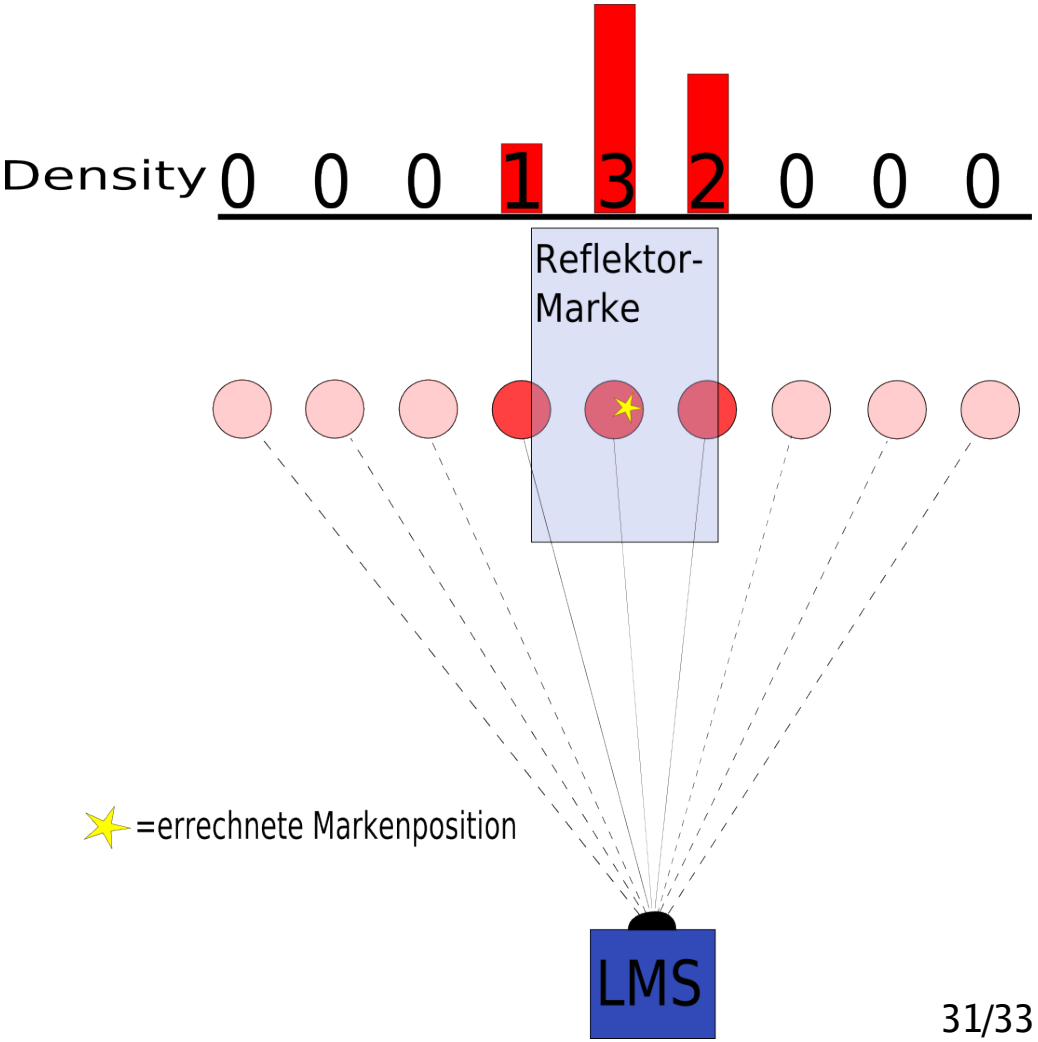
- 60cm Radius von Robotermitte
- 1m Radius von Robotermitte
- Visualisierung mittels LED-Zeile





Vorverarbeitung: Markenextraktion

- mitteln benachbarter Messwerte
  - Entfernung
  - Winkel
- nach Reflexionsstärke gewichtet
  - 3 Bit Auflösung



Benchmark-Ergebnisse

- Moxa-Karte
  - system 23,4 %
    - 14,4 % inb()
  - user 52,3 %
    - bei Verlust von 30-40 % der Messwerte
- Rabbit PowerCore 3800
  - system 3,9%
  - user 57,0%
    - ohne Verlust von Messwerten

Ergebnisse nicht mehr reproduzierbar, da die Lokalisierung mittlerweile mehr berechnet.

Ausblick

- Algorithmen für extrahierte Telegramme
  - Medianfilter
  - Reduktionsfilter
  - Linienfilter
  - Eckenfilter
  - ...
- gefundene Marken in Roboter-Koordinaten umrechnen

?