SEP SS07 / Lastenheft

Particle Analyzer in C/C++

Projekt:	Particle Analyzer in C/C++ 1.0
Autor:	Monty Beuster
Home:	http://www.ibr.cs.tu-bs.de/dus
letzte Änderung:	6. April 2007

Inhaltsverzeichnis

1 Zielbestimmungen

Welche Ziele sollen durch den Einsatz der Software erreicht werden?

Visualisierungstool für Sensornetzwerke

- **stabil**,
- **intuitive Bedienung**,
- **Erweiterbarkeit**.

Implementierung

- **C++**,
- **Qt-Oberfläche**

Ergebnis: Visualisierungstool als integraler Bestandteil der Particle-Computer-Platform.

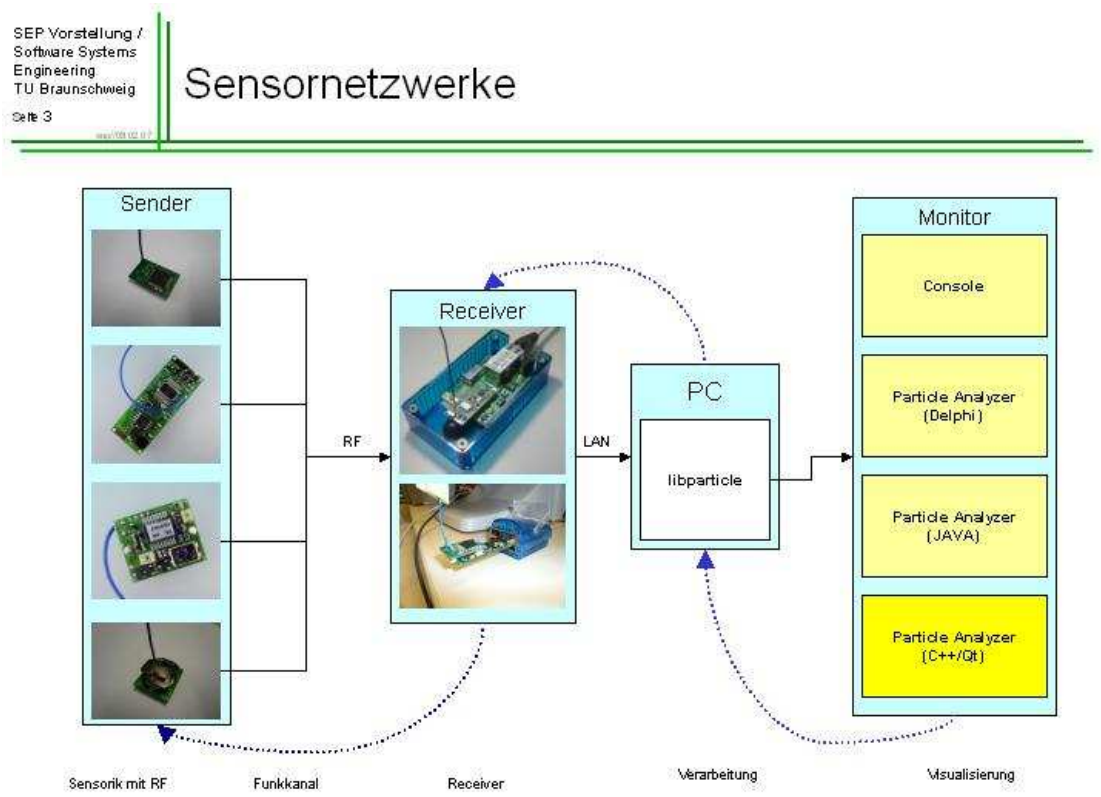

Abbildung 1: Übersicht Particle-Computer-Platform

2 Produkteinsatz

Für welche Anwendungsbereiche und Zielgruppen ist die Software vorgesehen?

Einsatzgebiete des Particle Analyzers sind in erster Linie Forschung und Lehre in den Bereich Ubiquitous Computing / Pervasive Computing.

Zugeschnitten auf die Particle-Computer-Platform soll es inbesondere Studenten, Studien- und Diplomarbeiter... etc. einen schnellen Einstieg in die Materie Sensorknoten ermöglichen. Daraus folgt eine klare Strukturierung der Benutzeroberfläche

Desweiteren soll die Erweiterbarkeit, insbesondere durch unerfahrene Programmierer, gewährleistet sein. Dies begründet eine klare Strukturierung des Quellcodes.

3 Produktfunktionen

Was sind die Hauptfunktionen des Produktes aus der Sicht des Auftraggebers?

3.1 Benutzerfunktionen

3.1.1 An- und Abmelden

Eine An- und Abmeldung im Sinne von Authentifizierung (Benutzername, Kennwort) ist nicht vorgesehen.

3.2 Das Minimalsystem

/LF0000/ Das Programm wird als executable oder in Form einer „one-click-installation“ ausgeliefert. Das Programm muss anschließend sofort lauffähig sein und darf keinerlei Abhängigkeiten (weitere dlls, Webserver oder Datenbanken) enthalten.

/LF0050/ Das Programm/Quellecode hat einen modularen Aufbau.

/LF0100/ Der Analyzer empfängt - auf beliebigen Port (default: 5555) ankommende Pakete. Dazu wird die Bibliothek *libparticle* der Particle-Computer-Platform bereitgestellt. Die Nutzung ist obligatorisch.

/LF0110/ Graphische Benutzeroberfläche und Visualisierung der Pakete mittels Qt/C++.

/LF0120/ Die Standartausgabe des Particle Analyzers, ist die Visualisierung der Pakete in einer Tabellenform.

/LF0130/ Analyzer sendet Broadcastpakete auf beliebigen Port (default: 5556).

/LF0140/ Um der Fülle an Informationen (Pakete), die über den Kanal eintreffen, gerecht zu werden stellt der Particle Analyzer verschiedene *Filterfunktionen* zur Verfügung. Die Filter sollen auf einfache Art und Weise im Benutzerinterface erstellt werden können. Es soll auch möglich sein mehrere Filter zu kombinieren. So sollen beispielsweise ausschließlich Pakete angezeigt werden, die...

- eine von einem bestimmten Particle stammen (Filtern nach ID)
- einen bestimmten ACL-Typen haben
- an einer beliebigen Stelle x in der Payload einen Wert annehmen/überschreite/unterschreiten... usw.

/LF0150/ Die Filterfunktionen des Analyzer können (optional) durch einen Signalflussgraphen erstellt, editiert und erweitert werden. So könnte man sich vorstellen, dass die Pakete zunächst nach ihrer ID und anschließend nach ihrem ACL-Typen gefiltert werden. Es entsteht somit eine Filterkaskade. Diese wird mit Qt-Technologie implementiert und kann beliebig erweitert, jederzeit gespeichert und geladen werden.

/LF0160/ Analyzer sendet adressierte Pakete auf beliebigen Port (default: 5556). Die Adresse besteht dabei aus acht Oktetts, die durch Punkte voneinander getrennt sind (Bsp: 1.2.3.4.5.6.7.8). Die pParts besitzen einen ID-Chip, während die ID bei cParts softwareseitig verändert werden kann.

Hinweise:

- Die IP eines Rechners mit der IP 192.168.1.1 würde dabei wie folgt lauten: 1.1.1.1.192.168.1.1
- cPart-IPs beginnen üblicherweise mit 1.2.3.4.x.x.x.x
- pPart-IPs beginnen i.d.R. mit 2.232.0.0.0.x.x.x

/LF0170/ Der Analyzer ist in der Lage Pakete acknowledged zu senden. D.h. der Analyzer sendet ein Paket und wartet auf eine Bestätigung (des Empfängers), dass das Paket angekommen ist (ACK). Erhält er in einer vorgegebenen Zeit kein ACK sendet er das Paket erneut. Nach einer frei vom Benutzer wählbaren Anzahl erfolgloser Versuche (Retrys) beendet der Analyzer den Sendeversuch.

/LF0180/ Der Analyzer ist in der Lage Pakete zu empfangen, deren Absender eine Bestätigung angefordert hat. Der Analyzer schickt ein Acknowledgement. Diese Funktion sucht man in den vorangegangenen Versionen des Particle Analyzers vergeblich.

/LF0190/ Der Analyzer beinhaltet eine Plot-Funktione, mit deren Hilfe verschiedenste Paketinhalte graphisch darstellen lassen. So könnte man sich beispielsweise den zeitlichen Verlauf eines Geschwindigkeitssensors o.ä. darstellen lassen. Der Benutzer muss dazu frei konfigurieren können, welchen Paketwert (Bsp. Payload an der Stelle 3) er geplottet haben möchte.

/LF0200/ Der Plot/Scope sollte sich (per Mausklick oder einfacher Benutzerführung) in herkömmliche Bildformate (jpg, gif, png...) exportieren lassen.

3.3 Persönliche Profile

/LF0210/ Der Benutzer kann eigene Profile speichern und laden. Dabei wird die zuletzt verwendete Konfiguration als default geladen. Dazu zählt insbesondere die Anordnung der Fenster (Senden, Empfangen, Filter, Plot... usw) und die zuletzt verwendeten Filter.

3.4 Erweiterte Funktionen

/LF0300/ ID in Liste aufnehmen Mit zunehmender Nutzungsdauer wünscht sich der Nutzer die Möglichkeit, eine bereits erkannte Particle-ID einfach wiederzuverwenden. Die Java-Version löst das Problem, in dem in einem Fenster alle gefundenen IDs aufgelistet werden. Ein Mausklick auf eine ID ermöglicht es, die ID zu kopieren und an einer beliebigen Stelle im Analyzer einzufügen (siehe Abb. 2). Diese Funktion fehlt beim Delphi-Analyter und ist in den C-Analyzer zu integrieren.

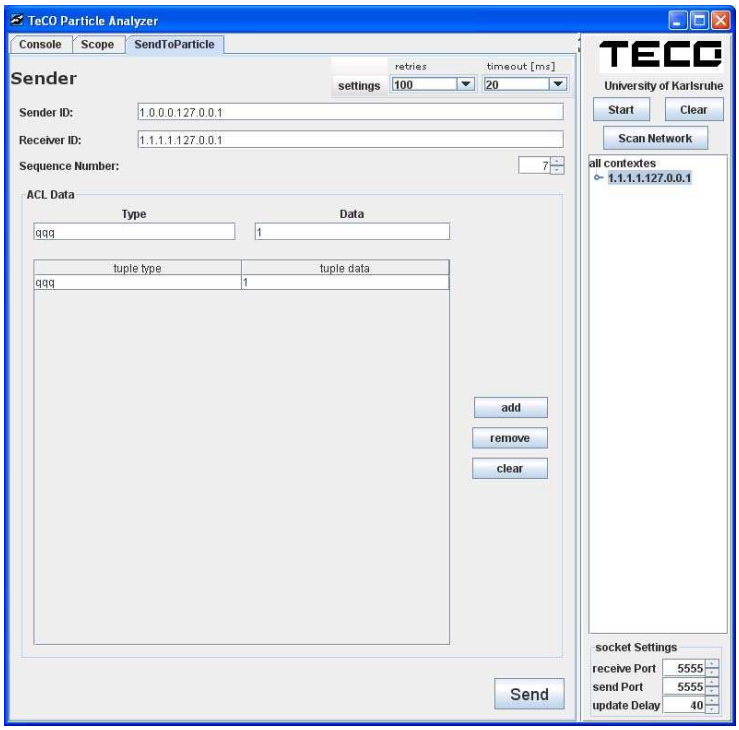

Abbildung 2: ID aus Liste in Sendeportfeld kopieren

/LF0310/ Traffic-Protokollierung Das System ist nicht nur in der Lage den Daten/Pakettransfer darzustellen. Optional sollen die Daten in eine Textdatei gespeichert werden können. Dabei wird dem Benutzer eine Standartausgabe vorgeschlagen. Der Nutzer soll aber in der Lage sein, die Daten(blöcke) frei zu wählen. So ist es zum Beispiel denkbar, dass der Benutzer lediglich an ID und Payload interessiert ist, während üblichweise eine Fülle von Daten exportiert wird.

/LF0310/ Replay von protokolliertem Traffic Die in einer Datei gespeicherten Daten sollen replayfähig sein, d.h. vorausgesetzt es stehen genug Informationen innerhalb der Datei zur Verfügung (siehe /LF 0310/) werden die Daten ins Netzwerk geleitet. D.h. eine einstmals durchgeführte Messung kann vollständig wiederholt (replayed) werden.

/LF0320/ In Datenbank schreiben Optional können Daten in eine Datenbank weitergeleitet werden. Der Benutzer wählt dabei Datenbankname, Tabellenname... usw.

/LF0330/ Aus Datenbank lesen Selbstverständlich können Daten, die in einer Datenbank abgelegt wurden, durch den Analyzer gelesen werden. Auch hierbei besteht die Möglichkeit ein Replay durchzuführen.

/LF0340/ Reguläre Ausdrücke Filter- und Rechenfunktionen sind durch reguläre Ausdrücke zu realisieren. Dabei kann der Benutzer beliebige reg. Ausdrücke eingeben.

Beispiele zur Anwendung dieser Funktion sind die mathematische Verarbeitung bestimmter Sensordaten oder die gesammelte Ausgabe aller Temperaturdaten usw.

/LF0350/ Verschiedene Sende-/Empfangsports in einem Analyzer Ein Analyzer soll in der Lage sein, auf verschiedenen Ports zu schreiben und zu hören. Diese Funktion ist als leztes zu realisieren.

4 Testroutinen

/LF0400/ Datenbank-Belastungstest Der Analyzer ist in der Lage, auch unter starken Belastungen, stabil zu arbeiten. Werden bspw. hunderte oder tausende Pakete in kürzester Zeit auf den Kanal geschickt, bleibt das System stabil und Echtzeitfähig.

Um diese Funktion zu testen, wird ein Belastungstesttool (ein spezieller Analyzer und eine Datenbank) zur Verfügung gestellt. Der Analyzer liest unter Angabe von Port und Datenbank, die vormals in der DB gespeicherten Daten und gibt sie mit gewünschtem Delay auf den Kanal aus. Wird der Delay sehr klein gewählt (wenige ms) wird eine hohe Belastung simuliert.

/LF0410/ μPart-Echtzeittest (Beschleunigungssensor) Auch unter hoher Last soll die Echtzeitfähigkeit erhalten bleiben. Um diese Funktion zu testen wird ein uPart bereitgestellt. Zusätzlich zu /LF 0400/ wird dieser, mit einem Beschleunigungssensor bestückte, cPart an das System angeschlossen. Der Analyzer wird auf die Filterung von Beschleunigungsdaten konfiguriert. Unter diese Belastung ist der Analyzer in der Lage, die Beschleunigungsveränderung in Echtzeit darzustellen. Ein ruckartiges Bewegen des uPart führt eine umgehenden Darstellung der Daten mit sich.

5 Ergänzungen

Gibt es noch aussergewöhnliche Anforderungen?

5.1 Realisierung

Das vorliegende System muss mit **C++** und **Qt** realisiert werden.

5.2 Ausblick auf die nächste Version

In kommenden Versionen werden spezifische Erweiterungen und Änderungen implementiert, die sich aus den Erfahrungen und Wünsche der Anwender ergeben. Dabei sind sowohl Änderungen als auch neue Module denkbar. Demzufolge muss der Analyzer so realisiert werden, dass beides leicht realisierbar ist.

5.3 Nützliche Links

SEP-Page ...

Howto cPart ...

Delphi-Particle Analyzer ...

Console-Particle Analyzer ...

Java-Particle Analyzer ...