Studienarbeit I

Fernüberwachung und Fernsteuerung über Internet (Android Applikation)

Studiengang Informationstechnik

an der Dualen Hochschule Baden-Württemberg Stuttgart

****

von

Hans Joachim Krauch

und

Waldemar Siebert

Kurs TIT09IN

Betreuer Klaus Hartmann

Inhaltsverzeichnis

[Aufgabenstellung 3](#_Toc317368637)

[Warum wir uns für den Net-IO entschieden haben 4](#_Toc317368638)

[Warum wir uns für Ethersex entschieden haben 5](#_Toc317368639)

[Zusammenspiel Android-App, Ethersex und Steuercontroller 6](#_Toc317368640)

[Stand der Dinge 7](#_Toc317368641)

[App-Design, Webserver-Design 8](#_Toc317368642)

[Datenverwaltung 9](#_Toc317368643)

# Aufgabenstellung

Thema: Fernüberwachung eines Mikrokontrollers

Quellen:

* AVLFreaks
* U. Radig
* <http://www.androidpit.de/de/android/wiki/view/Android_Anf%C3%A4nger_Workshop>

Beispielvorgehen:

1. Kontroller + AVR Architektur + Code verstehen
2. Kontroller wählen und kaufen (40Pin)
3. TCP/IP Stack implementieren 🡪 AVR Webserver
4. Android App: feste Felderanzahl, Messwerte sind irrelevant
5. Daten kommen von einem zweite Kontroller über SPI
6. Soll über das Internet ansprechbar sein (Auslesen und Schalter)
7. Schalter soll dargestellt werden, erst wenn auch umgeschaltet wurde soll die Stellung verändert werden

Ziel der Dokumentation:

* Ist eher für die Wartung 🡪 Modulbeschreibung, Aufbau, usw.
* Bedienungsanleitung
* Installationsanleitung

# Hardwareauswahl

Für die Umsetzung eines Mikrokontrollers, der mit einem Ethernet Controller kommuniziert kamen mehrere Möglichkeiten in Frage. Bei einer Recherche wurde ermittelt, dass es bereits ähnliche Projekte gibt, sodass hierfür entweder die Dokumentation von U. Radig hinzugezogen werden konnte, oder aber der kommerzielle Selbsbaukasten NET-IO von Pollin, der sowohl einen Ethernet Controller und Anschluss, als auch einen RS232 Controller und Anschluss hat.

Hauptgrund für die Verwendung des NET-IO ist der gute Support durch die verwendete Ethersex-Firmware. Zudem ist der Bausatz recht güngstig und lässt sich mit etwas Lötkenntnissen schnell aufbauen. Des Weiteren besitzt das NET-IO eine ISP-Schnittstelle, die dazu genutzt werden kann, den verwendeten Controller direkt auf dem Board zu programmieren.

# Warum wir uns für Ethersex entschieden haben

Bei der Recherche nach einer geeigneten Firmware für den Mikrokontroller, sind wir auf das „Ethersex“-Projekt gestoßen. Ethersex ist eine Firmware, welche für die Verwendung auf AVR 8-Bit Mikrokontroller mit Netzwerkanschluss zugeschnitten ist.

Das Ethersex-Projekt unterstützt mehrere Boards darunter das AVR NET-IO von Pollin, für welches wir uns entschieden haben.

Die Firmware zeichnet sich durch einen bereits implementierten TCP/IP-Stack (IPv4 & IPv6) und zahlreiche weitere Features, wie z.B. ein HTTP-Server, aus.

Auf dem Mikrokontroller sorgt Ethersex für die Kommunikation mit dem Ethernet-Controller und die Verarbeitung der IP-Pakete. Ethersex erlaubt es außerdem, was für unser Vorhaben äußerst nützlich ist, TCP-Befehle (ECMD) an den Mikrokontroller zu schicken. So kann bspw. Der Hostname des Mikrokontroller an einem PC abgefragt werden. Näheres zu Ethersex siehe <http://old.ethersex.de/index.php/Ethersex-Artikel>.

# Zusammenspiel Android-App, Ethersex und Steuercontroller

Per Android-App wird ein bestimmter TCP-Befehl an den am Netz angeschlossenen Ethersex-Controller gesendet. Der Ethersex-Controller reagiert auf den Befehl und empfängt die übergebenen Parameter. Ohne diese zu verarbeiten, leitet er diese an den über SPI angeschlossenen Steuercontroller weiter.

Im Steuercontroller wird der Befehl verarbeitet und der Rückgabewert anschließend wieder über die SPI-Schnittstelle dem Ethersex-Controller mitgeteilt. Dieser packt den Rückgabewert in ein bzw. mehrere IP-Pakete und sendet die Pakete dem Android-App als Antwort zurück.

Der Ethersex-Controller verhält sich im Grunde wie ein Proxy-Server: Er empfängt einen Request und leitet diesen an den Steuercontroller weiter. Die Antwort vom Steuercontroller wird ebenso an die Android-App weitergeleitet.



Abbildung 1: Client-Proxy-Server Konzept

@WALDE: In MS Word die Abbildung 6.1 aus dem Linkohr sein Script (S. 18) als Smart-Art darstellen. Client=Android-App, Proxy=Ethersex-Controller, Server=Steuercontroller

# Stand der Dinge

**Ethersex**:

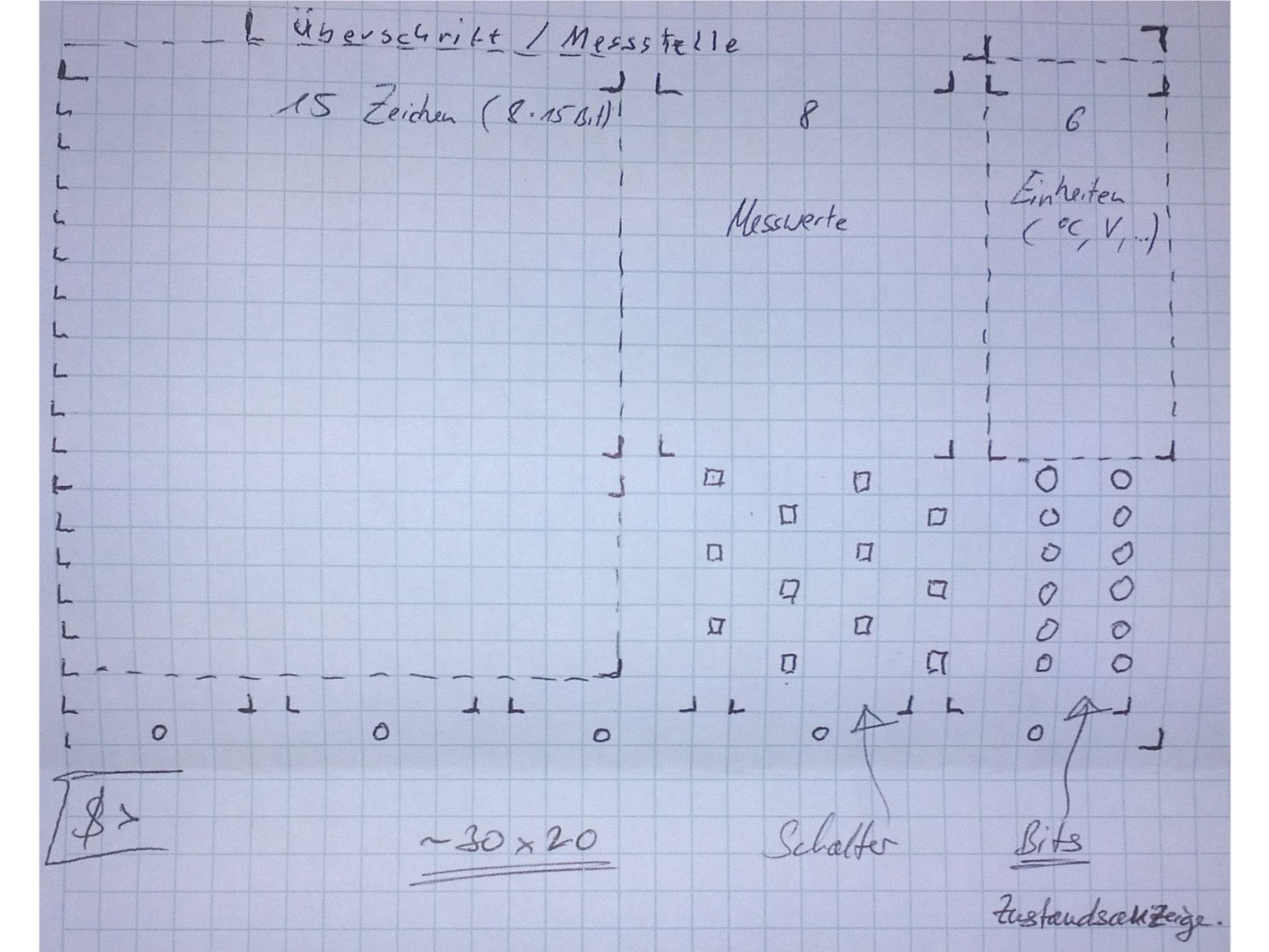
Das AVR NET-IO wurde erfolgreich aufgebaut und mit einem ATMEGA 644 bestückt (ausreichend Speicherkapazität). Die Ethersex-Firmware wurde angepasst, kompiliert und auf den Mikrokontroller übertragen. Mit dem Webbrowser ist es möglich den Webserver des Ethersex-Controllers aufzurufen. ECMD-Befehle (Ethersex-Commands) konnten erfolgreich über TCP an den Mikrokontroller gesendet werden.

**Android-App:**

Die Verbindung einen Java Socket über Android konnte noch nicht durchgeführt werden und das App Design ist auch noch in Arbeit.

**@Walde**

# App-Design, Webserver-Design



1. Textwert dann Messwert dann Einheit als eine Zeile, **in Folge geschickt**.

ASCII Zeile wird ausgelesen, Spalte wird übersprungen bei def. Sonderzeichen.

# Datenverwaltung

Wie die Daten geholt werden, ob Zwischenspeichern auf dem Controller und wann von dem andern Controller ler geholt werden will. Nur bei Zugriff, oder mit Aktualisierungstaste.

Bei Zugriff und wenn auf Aktualisierungs-Taste gedrückt wird.

→ Kein Zwischenspeichern auf dem Controller