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](#_Toc317427526)

[Auswahl des Mikrokontrollers 4](#_Toc317427527)

[AVR-NET-IO 5](#_Toc317427528)

[Firmware 6](#_Toc317427529)

[Kommunikationskonzept 7](#_Toc317427530)

[Stand der Dinge 8](#_Toc317427531)

[App-Design, Webserver-Design 9](#_Toc317427532)

[Datenverwaltung 10](#_Toc317427533)

# Aufgabenstellung

Thema: Fernüberwachung eines Mikrokontrollers

Quellen:

* AVLFreaks
* U. Radig
* <http://www.androidpit.de/de/android/wiki/view/Android_Anf%C3%A4nger_Workshop>
* [http://www.ethersex.de](http://www.ethersex.de/)
* <http://www.mikrocontroller.net/articles/AVR_Net-IO_Bausatz_von_Pollin>
* Bild:http://www.pollin.de/shop/dt/MTQ5OTgxOTk-/Bausaetze\_Module/Bausaetze/Bausatz\_AVR\_NET\_IO.html

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

1. Auswahl des Mikrokontrollers

Für die Umsetzung eines Mikrokontrollers, der mit einem Ethernet Controller kommuniziert kamen mehrere Möglichkeiten in Frage. Zum einen war es möglich alles neu zu entwickeln und hierfür einen Mikrokontroller nach Vorgaben auszuwählen, oder aber auf bereits bestehende Projekte aufzubauen. Die Vorgaben hierfür waren, einen AVR mit 40 Beinen zu nutzen, der mit einem Ethernet Controller kommunizieren soll und genügend Platz für einen Webserver aufweist. Durch eine Recherche sind zwei Projekte aufgefallen. Dabei handelt es sich um einen Webserver von Ulrich Radig, der eine Dokumentation inklusive Schaltplan bieten, oder aber der kommerzielle Bausatz AVR-NET-IO von Pollin, welcher inklusive aller Bauteile und einer kleinen Software bestellt werden kann. Eine Bestellung als Fertigmodul ist auch möglich, ist aber entsprechend teurer.

Die Auswahl fiel hierbei auf den AVR-NET-IO von Pollin. Der Hauptgrund für die Verwendung des NET-IO ist der gute Support durch die verwendete Ethersex-Firmware. Zudem ist der Bausatz recht günstig und lässt sich mit den nötigen Lötkenntnissen und etwas Aufwand zusammenbauen. Für die Kommunikation zwischen dem Mikrokontroller und dem Ethernet Controller wird SPI verwendet. Desweiteren besitzt das NET-IO eine ISP-Schnittstelle, mit der der verwendete Controller direkt auf dem Board programmiert werden kann.

1. AVR-NET-IO

Der AVR-NET-IO besteht aus einer Ethernet-Platine mit einem Sockel für einen 40 Pin Mikrokontroller, einem Netzwerkcontroller ENC28J60 und einem MAX232 für die serielle Schnittstelle. Auf der Platine befinden sich 8 digitale Ausgänge, 4 digitale Eingänge und 4 ADC-Eingänge, die über TCP/IP abgerufen/geschaltet werden können. Desweiteren ist eine RJ45 Netzwerkbuchse, ein 9-poliger Sub-D Anschluss für RS232, eine ISP-Schnittstelle und alle weiteren benötigten Komponenten vorhanden. Die Platine muss mit 9V Betriebsspannung versorgt werden und hat eine Stromaufnahme von etwa 190mA. Die digitalen Ein- und Ausgänge sind mit 0V und 5V als Richtwert angegeben. Die ADCs des Mikrokontrollers haben eine Genauigkeit von 10Bit. Die Platine hat mit den verwendeten Komponenten die Maße 108mm Länge, 76mm Breite und 22mm Höhe.

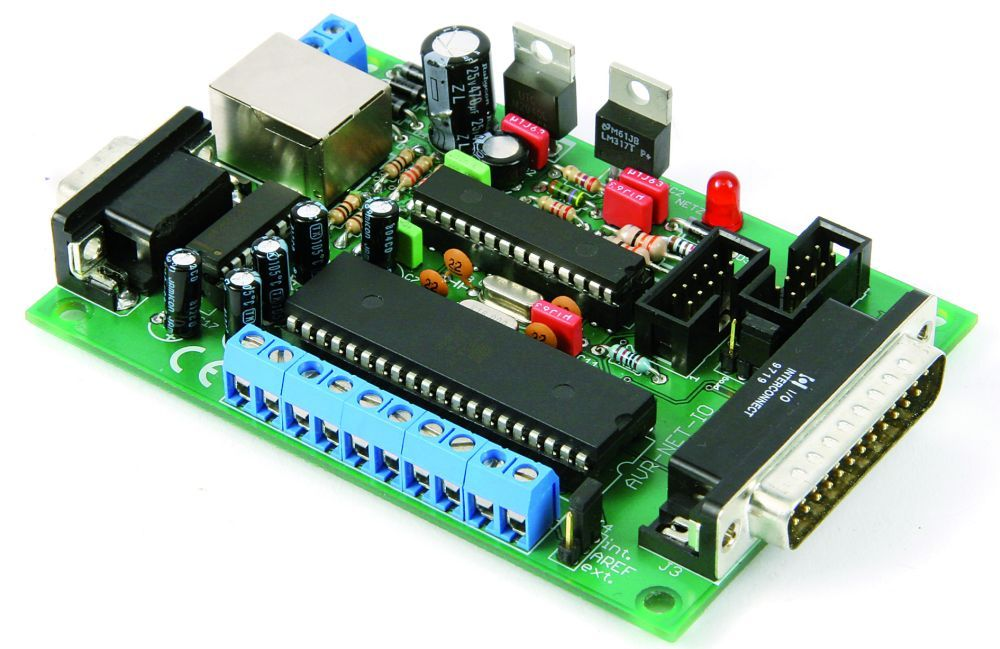


Abbildung 1: AVR-NET-IO von Pollin

# Firmware

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>.

# Kommunikationskonzept

Das Kommunikationskonzept besteht aus mehreren Komponenten. Es muss eine Kommunikation zwischen einem Android Smartphone und dem Mikrokontroller aufgebaut werden. Hierfür wird per Android-App 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 diesen in ein bzw. mehrere IP-Pakete und sendet diese 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 2: 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

Der Mikrokontroller soll per Android-App über das Internet angesprochen werden. Daraufhin soll er die Anfrage an einen Kontroller weiterleiten, der Messwerte erfasst. Die Messwerte sollen dann über den für die Kommunikation zuständigen Mikrokontroller an die Android-App weitergeleitet werden.

# Stand der Dinge

**Ethersex**:

Das AVR NET-IO wurde erfolgreich aufgebaut und mit einem ATmega644 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.

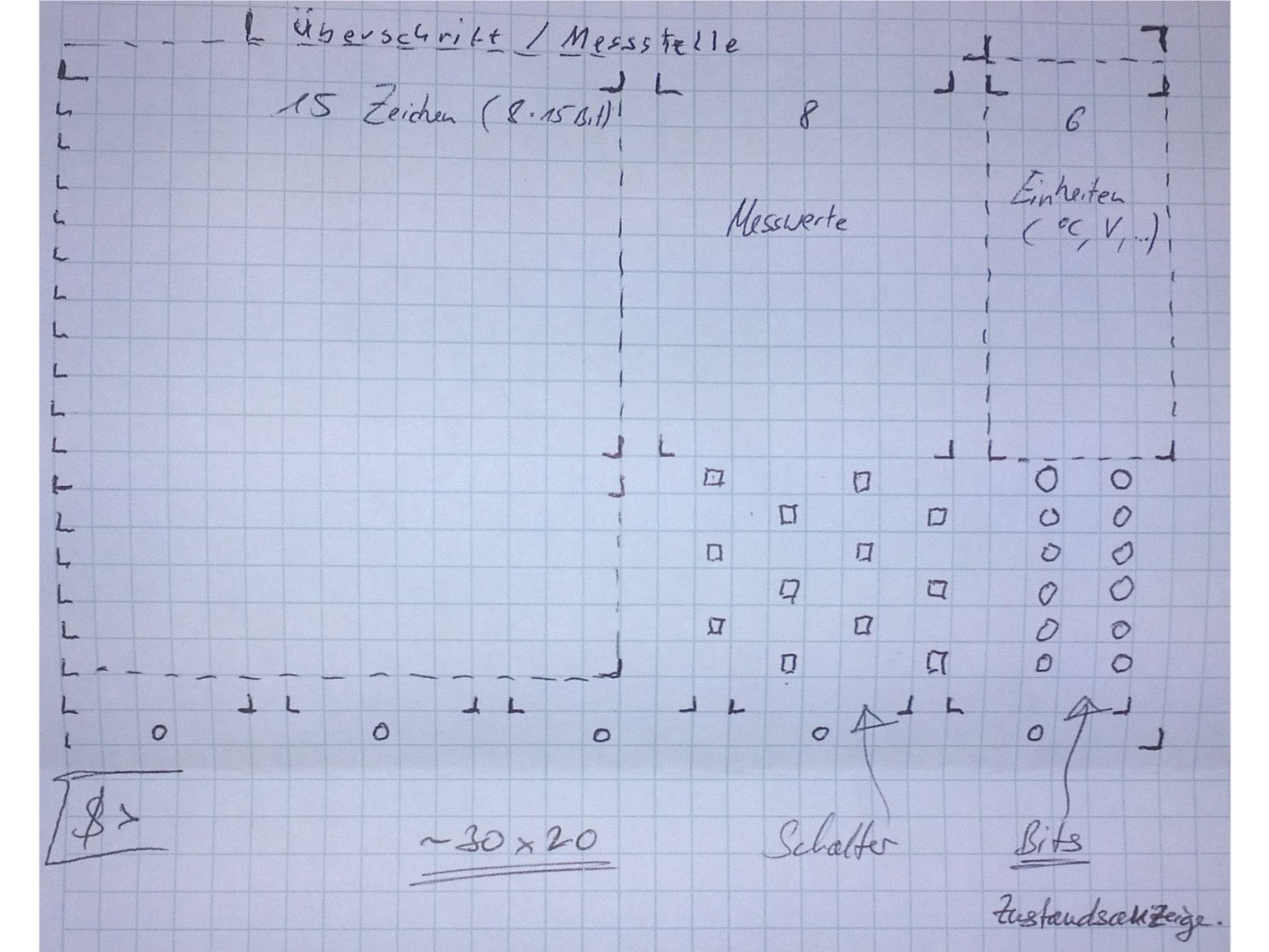
**Android-App:**

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

Die Grafische Oberfläche ist in der Mache und die Übertragung von TCP Befehlen auch.

**@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