Ethernet I2C Server

Alex Hirsch Felix Kostenzer

2010 / 2011

Inhaltsverzeichnis

Kapitel 1

Aufgabenstellung

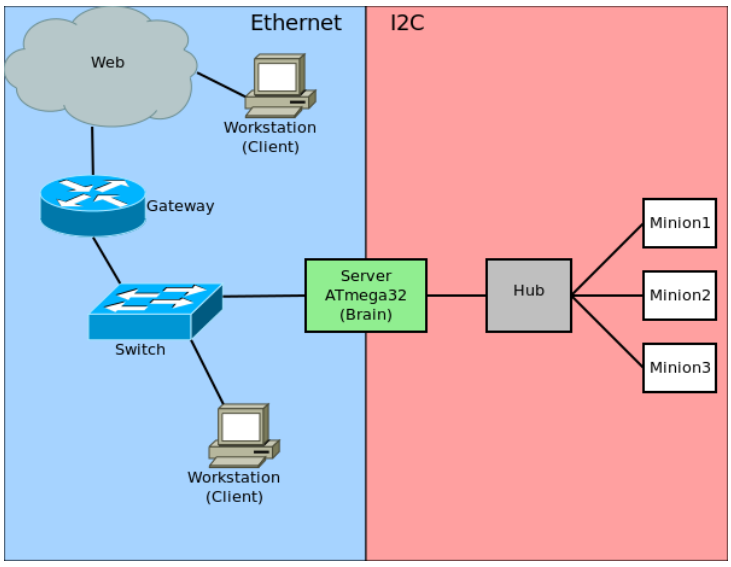
1.1 Beschreibung

Dieses Projekt wird im Zuge des FTKL Unterrichts der 5. Klasse durchgeführt. Entwickelt wird ein elektronisches Steuersystems für Häus.

Die Elemente des Systems (Sensoren, Motoren, ...) sollen (über Netzwerk) von einem Computer angesteuert werden können. Da es kompliziert wäre jedes dieser Elemente mit einer Ethernet-Schnittstelle zu versehen (und somit das gesamte Ethernet-Handling jedem Element zu überlassen) wird ein Gerät entwickelt, das als Ädapterßwischen Ethernet und I2C Bus dient. Dieses Gerät wird folglich als Ëthernet I2C Server"bezeichnet.

Der Computer, der das System ansteuert, wird als Client bezeichnet. Um Konflikte mit den steuernden Elementen wie Sensoren, Motoren, ... zu verhindern, werden diese Elemente "Minions"getauft.

1.2 Konzept



Diese Grafik beschreibt den finalen Aufbau. Einer der Clients befindet sich im LAN, der andere im Internet (bzw. WAN). Der Zugriff auf den Server ist von beiden Clients aus möglich, sofern der Router (Gateway) die entsprechende NAT / PAT Konfiguration besitzt.

Die Minions sind mit dem Server über einen HUB verbunden. Dieser HUB dient als Verbindung um die Minions an den Bus anzukuppeln. Hier können die benötigten Pull-Up-Widerständen angebracht weden.

1.3 Hinweis

Dieses Projekt soll als Prototyp für den nächsten Jahrgang dienen. Daher geht es hier Hauptsächlich um "Prove of Concept".

Da andere Schüler auf diesem Projekt aufbauen sollen, wird versucht den Quellcode so übersichtlich und einfach wie möglich zu halten. Es werden daher nur die notwendigsten Segmente implementiert um Übersichtlichkeit zu gewährleisten.

Es wird keine Platine gefertigt, da die Anforderungen für die finale Version noch unklar sind. Als Hardware für den Prototypen dient uns das NetIO Board (v1.0). Jedoch gibt es einen ersten Ansatz für die Platine. Die Abgabe des Projektes wird die (unfertigen) Files der Platine beinhalten, sodass auf diesen Aufgebaut werden kann.

1.4 Ziel

Als finales Ziel wird ein fehlerfreier Kommunikations-Ablauf festgelegt. Dazu wird an einem Minion ein Port (8 Bit) als Ausgang und ein Port (8 Bit) als Eingang verwendet. Der Client soll die 8 Bits des einen Ports setzen (bzw. löschen) können, sowie die 8 Bit des anderen Ports auslesen können. (Jedes Bit soll einzeln gelesen bzw. gesetzt / gelöscht werden können)

Der (Test-)Minion wird ebenfalls mit einem ATmega32 aufgebaut.

1.5 Meilensteine

- I2C Kommunikation funktioniert bidirektional
- Ethernet Kommunikation funktioniert bidirektional
- Der gesamte Kommunikations Ablauf funktioniert fehlerfrei mit eigener Software
- Der gesamte Kommunikations Ablauf funktioniert fehlerfrei mit der Software der anderen Gruppe

Kapitel 2

Planung

2.1 Server

Das NetIO Board verwendet einen ATmega32 Controller, welcher über das ISP Interface programmiert werden kann. Die Ethernet-Schnittstelle wird über einen Ethernet Chip (enc28j60) mit dem SPI Interface des Controllers verbunden. Als Debug Interface dient die USART Schnittstelle.

Das I2C Interface wird über einen 25 poligen Sub-D Stecker heraus geführt.

2.2 Test-Minion

Der Test-Minion wird an den I2C Bus des Servers angefügt.

Als Hardware wird das HTL Board (v2.3) verwendet. Benützt werden der ADC und ein Port des Controllers (ebenfalls ATmega32). Dabei werden entweder der aktuelle Wert am ADC Eingang ausgelesen, oder die 8 LEDs auf dem HTL Board angesteuert.

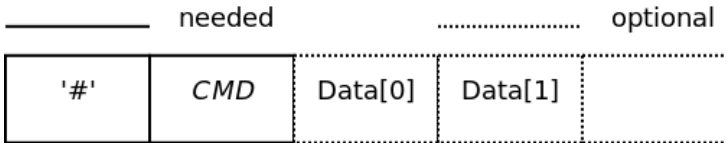
2.3 Protokolle

2.3.1 UDP

Da die Verbindung vom Client zum Server über Ethernet läuft, gibt es verschiedene Protokolle, die zur Datenübertragung verwendet werden können. Das UDP Protokoll bietet eine einfache Möglichkeit die Aufgabenstellung zu erfüllen. Moderne Programmiersprachen verfügen über eine Socket API mit denen Daten sehr komfortable über UDP gesendet und empfangen werden können. Da das UDP Protokoll sehr einfach aufgebaut ist, lässt es sich leicht in das Programm des Mikrocontrollers implementieren.

2.3.2 Daten-Struktur

Für die Daten muss jedoch eine Struktur festgelegt werden, damit der Server weißan welchen Minion er Daten schicken, bzw. von welchem Minion er Daten anfordern muss.



Um zu versichern, dass das erhaltene Paket wirklich eine Anweisung für unseren Server ist, wird für Anweisungen das Start-Symbol '#' festgelegt. (Dies wird laut ASCII Tabelle das erste Byte des Daten-Paketes sein. Es folgt ein 1 Byte großer Wert, der angibt, welche Anweisung ausgeführt werden soll. Im Speicher des Server befindet sich eine Tabelle, in der weitere Informationen stehen. Alles was hinter dem Änweisung-Byteßteht, wird als Daten behandelt. Stimmt die Größe dieser Daten mit der in der Tabelle festgelegten Daten-Größe überein, wird die Anweisung ausgeführt.

Jede Anweisung (egal ob gesendet oder empfangen werden soll) hat eine festgelegte Datengröße. Die Anzahl der Datenbytes muss eingehalten werden. Fragt der Client also Daten vom Server ab, muss er die entsprechende Anzahl an Datenbytes bei der Anfrage mit schicken (Der Inhalt dieser Datenbytes wird vom Server mit den gewünschten Daten überschrieben). Anschlies send werden die Adressen (Source bzw. Destination) ausgetauscht und das Paket wird an den Client zurück geschickt, sofern dieser Informationen abgefragt hat

2.3.3 Command-Table

Es wird ein Command Table festgelegt, in welchem zusätzliche Informationen zu einem Command (Anweisung) zu finden sind. Dieser Table wird sowohl beim Client, als auch beim Server benötigt, da beide wissen müssen, welche Datengrö e ein bestimmter Command erfordert.

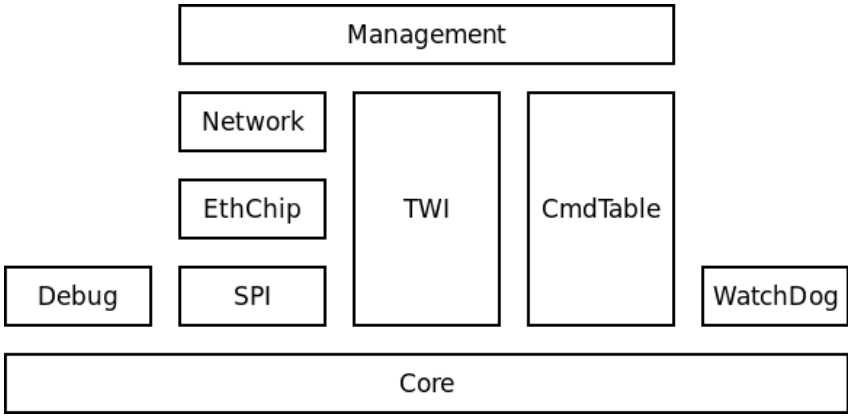
Darüber hinaus muss der Server wissen, welchen Minion er ansprechen muss, und ob er Daten von diesem Minion abfragen soll, oder dem Minion Daten schicken soll.

CMD	Minion Address	R / W	Data Size

Kapitel 3

Entwicklung

Dieses Projekt besteht grundsätzlich aus mehreren Modulen, welche parallel von einander entwickelt werden. Es werden jedoch zu Beginn die Komponenten implementiert, welche die weitergehende Entwicklung vereinfachen.



Die oberen Module hängen von den unteren Modulen ab, jedoch ist es möglich Module auszutauschen oder zu modifizieren, sofern die Anbindungen an die restlichen Module unverändert bleibt. (Oder diese Module ebenfalls entsprechend modifiziert werden.)

3.1 Debug

Um ein einfaches Debuggen zu ermöglichen wird die printf-Funktion implementiert, sodass Informationen über die USART an ein Terminal übertragen werden kann

3.2 TWI (=I2C)

Das Two-Wire Interface bietet eine komfortable Möglichkeit Daten seriell von einem Controller zum anderen zu Übertragen. Betrieben wird die TWI im Single-Master-Multi-Slave Modus. Die Slave-Controller können eigenständig keine Daten an den Master übertragen.

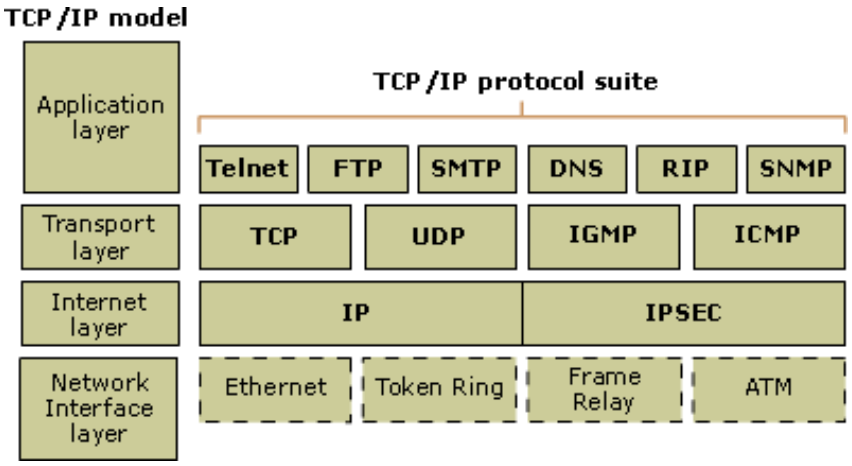
Für dieses Projekt werden nur 2 Funktionen entwickelt. Eine um ein Byte an einen beliebigen Teilnehmer zu senden und eine um ein Byte eines beliebigen Teilnehmers abzufragen. In diesen beiden Funktion ist ein kompletter Handshake implementiert. Sie reichen aus um die gesetzten Ziele zu erreichen. Es kann ohne weiteres auf diesen Function aufgebaut werden, um eine beliebige Anzahl an Bytes zu senden / empfangen.

Die Genauere Funktionsweise der TWI kann dem Datenblatt des Controllers entnommen werden.

3.3 ENC28J60

Dies ist der Chip, welcher Pakete über das Netzwerk Interface an den Controller weiter gibt. Er ist mit dem Controller über das SPI Interface verbunden. Anzumerken ist hier, dass der Chip lediglich Pakete austauscht. Er selbst bearbeitet die Pakete nicht. Es müssen also die notwendigen Protokolle am Controller implementiert werden. Bei der Ansteuerung des Chips hilft ein Projekt namens Etherrape (by Alexander Neumann), welches unter der GNU GPL Lizenz zur Verfügung steht. `http://www.lochraster.org/etherrape`

3.4 Ethernet



In dieser Grafik wird links das TCP/IP Modell gezeigt und rechts die Protokolle welche in den jeweiligen Ebenen verwendet werden. Ganz unten findet sich das Ethernet Protokoll in welchem die Adressierung über MAC-Adressen statt findet. Hier wird auch der Typ des darauf aufbauenden Protokolles angegeben. Auf dem Ethernet Protokolle bauen ARP (Address Resolution Protocol) und IP (Internet Protocol) auf. Auf IP bauen wiederum ICMP und UDP auf.

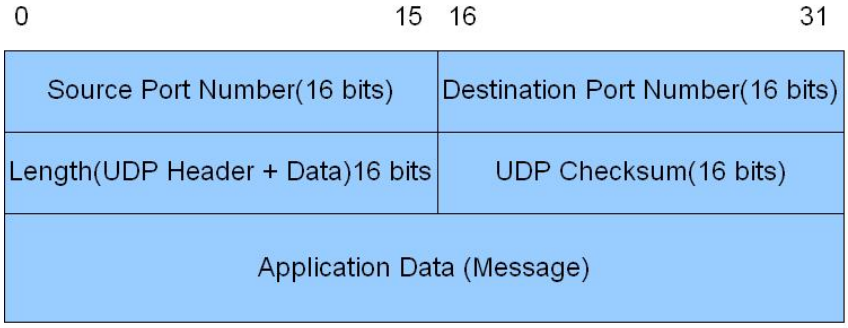
3.4.1 ARP (Address Resolution Protocol)

Das Address Resolution Protocol verbindet die MAC-Adressen mit den entsprechenden IP-Adressen. Bei einer Datenübertragung werden sowohl MAC als auch IP benötigt. Da die logische Adressierung (IP) bekannt ist, muss die MAC-Adresse erst abgefragt werden (ARP Request). Der Server muss hier mit seiner MAC Adresse antworten (ARP Reply).

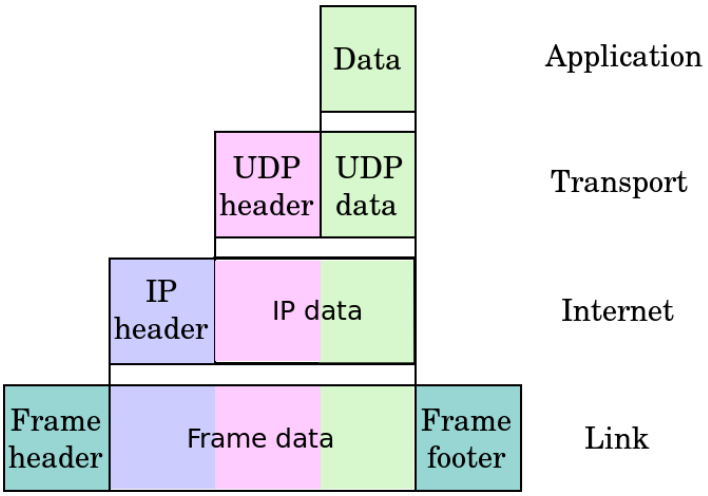
3.4.2 ICMP (Internet Control Message Protocol)

Es wird nur ein kleiner Teil des ICMP Protokolls implementiert, der Ping-Reply-Teil. Es soll möglich sein den Server anhand seiner IP Adresse zu pingen um die Verbindung zu überprüfen.

3.4.3 UDP (User Datagram Protocol)

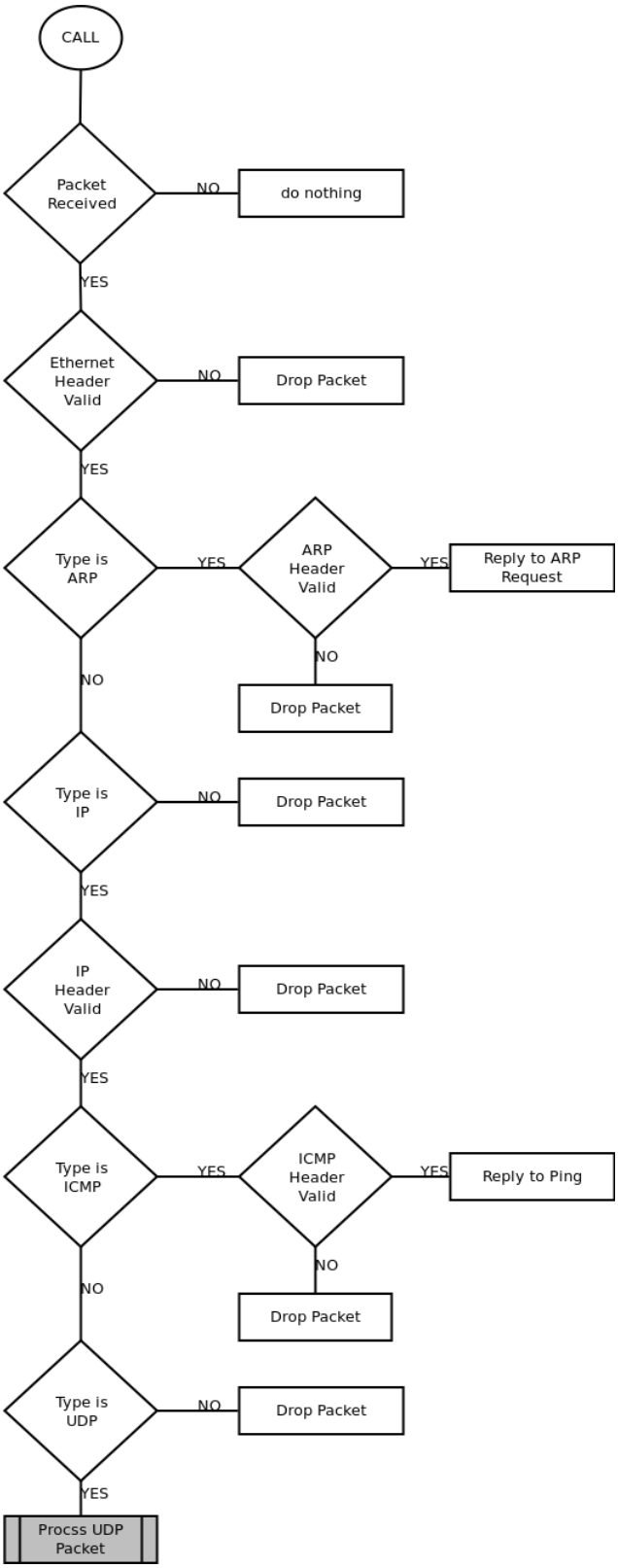


Der UDP Header besteht aus Source Port, Destination Port, Längen-Angabe (über Header und Daten), UDP Checksum. Nach dem Header folgen die Daten.



Diese Grafik verdeutlicht wie die Daten über Ethernet gesendet werden. Die Daten, welche unser System steuern (bzw. Informationen abfragen) befinden sich im UDP Paket.

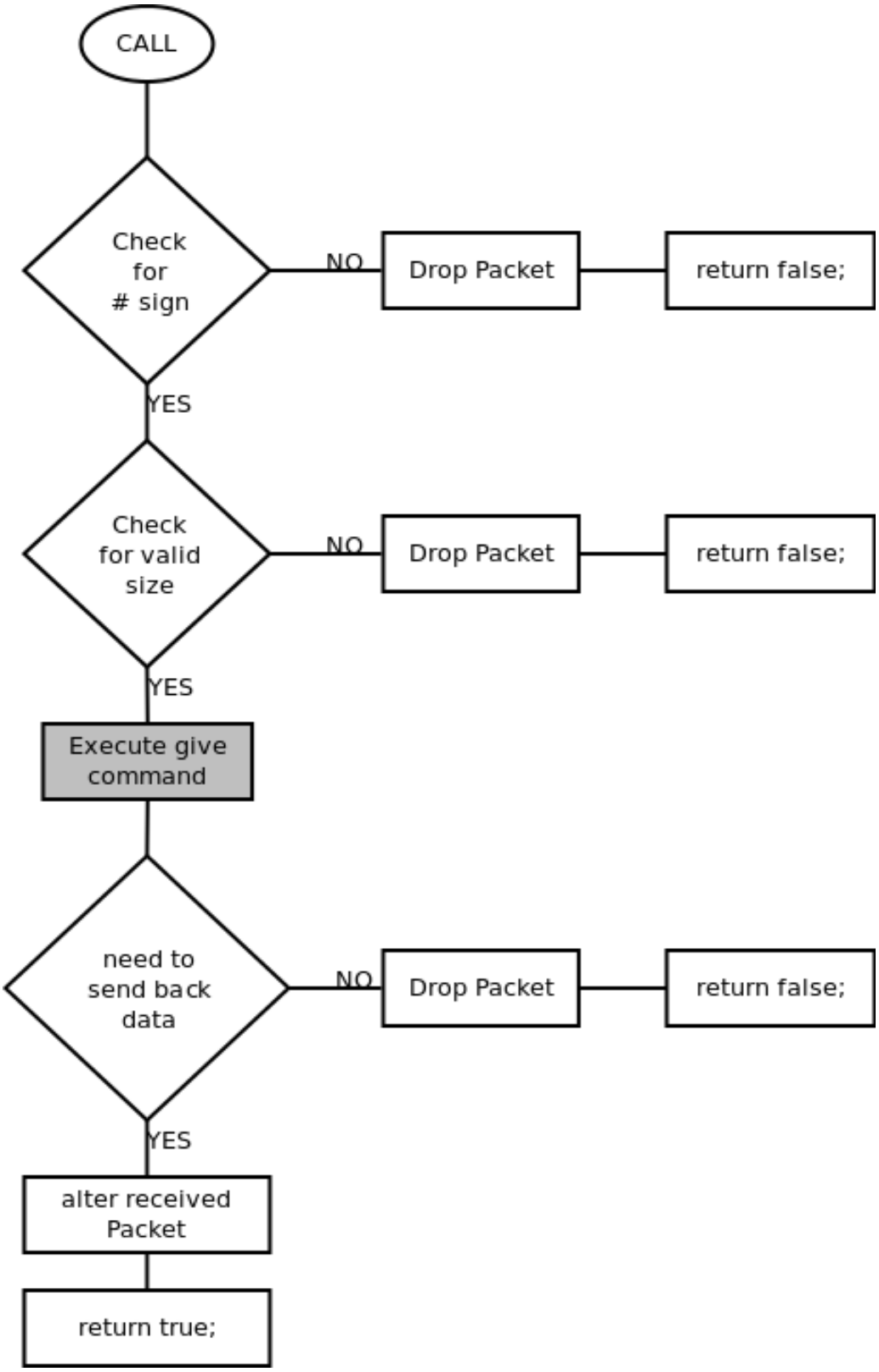
3.4.4 Process


CALL
Packet Received
NO
do nothing
YES
Ethernet Header Valid
NO
Drop Packet
YES
Type is ARP
YES
ARP Header Valid
YES
Reply to ARP Request
NO
Drop Packet
NO
Type is IP
NO
Drop Packet
YES
IP Header Valid
NO
Drop Packet
YES
Type is ICMP
YES
ICMP Header Valid
YES
Reply to Ping
NO
Drop Packet
NO
Type is UDP
NO
Drop Packet
YES
Procss UDP Packet

Dieser Flowchart zeigt den Ablauf der Ethernet Process Routine. Es wird überprüft, ob ein Paket empfangen wurde und folglich welche Reaktion ausgeführt werden soll. Durchläuft das Program die gesamte Baumstruktur bedeutet dies, dass ein gültiges UDP Paket empfangen wurde. Es wird ein Callout ausgeführt, in welchem der Inhalt des UDP Paketes verarbeitet wird.

3.5 Management

3.5.1 Process



Diese Routine wird durchgeführt wenn ein gültiges UDP Paket empfangen wurde (Callout vom Ethernet Process). Es wird geprüft ob der Inhalt dieses Paket einer gültigen Anweisung entspricht (Command Check, Data Size Check). Ist der Inhalt eine gültige Anweisung, wird diese ausgeführt und ggf. werden Informationen an den Client zurück gesendet, dies wird über den Return-Wert dieser Funktionen geregelt.

3.5.2 Watch-Dog

Es wird zusätzlich noch der Watch-Dog-Timer (WDT) aktiviert und auf 2 Sekunden gestellt. Sollte der Server das Paket also nicht innerhalb von 2 Sekunden abgearbeitet haben (weil ein Fehler aufgetreten ist), wird dieser neu initialisiert. Es wird über das Debug-Interface eine entsprechende Information ausgegeben, dass der Watch-Dog-Timer ausgelöst hat.

Kapitel 4

Test Minion

Um die Funktion des Servers überprüfen zu können, wird ein Test-Minion mithilfe des HTL Boards gebaut. Hier soll der aktuelle Wert des ADC ausgelesen werden können und über die TWI abgefragt werden. Zudem soll man die 8 LEDs auf dem HTL Board ansteuern können.

4.1 TWI

Empfängt die TWI Peripherie des Test-Minion etwas über das TWI-Interface, wird ein Interrupt ausgelöst, in welchem anhand des aktuellen Status eine Reaktion durchgeführt wird. Es werden lediglich die wichtigen Reaktionen aus programmiert. Error handling wurde nicht implementiert.

4.2 Watch-Dog

Wie beim Server wird der Watch-Dog-Timer aktiviert und auf 2 Sekunden gestellt. Tritt ein Fehler auf, sodass der Controller nicht mehr reagiert, wird ein Reset ausgelöst.

Anhang A

Misc

A.1 License

```
Copyright (C) 2010 / 2011  Hirsch & Kostenzer (06A HELI)
Hirsch: ax.warhawk@gmail.com
Kostenzer: fkostenzer@live.at

enc28j60 code based on etherrape (C) by Alexander Neumann
email: alexander@bumpern.de
url: http://www.lochraster.org/etherrape

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

A.2 GIT

Für die Entwicklung dieses Projektes wurde ein Version Control System (GIT) zur Hilfe gezogen. GIT managed die verschiedenen Versionen des Quellcodes während der Entwicklung. Es stellt zusätzlich noch viele weitere Features zur Verfügung, weitere Informationen unter `http://git-scm.com/`.

Das Projekt steht unter `http://github.com/W4RH4WK/06AHELI_FTKL_SERVER` zur Verfügung.

A.3 Doxygen

Es wurde zusätzlich Doxygen verwendete um es anderen Leuten zu erleichtern mit dem von uns entwickelten Programm zu arbeiten. Dazu wurde ein im src Ordner ein Doxygen Ordner erstellt, welcher ein Konfigurationsfile für Doxygen beinhaltet. Als Output-Format wurde HTML verwendet, da dieses unserer Meinung nach die komfortabelste Methode ist, um den Source-Code zu durchsuchen.

Weitere Informationen unter `http://doxygen.org`