Balthasar-Neumann-Technikum
Paulinstr. 105
54292 Trier

Projektdokumentation

Lernfähige Infrarotfernbedienung mit webbasierender Bedienoberfläche

Benjamin Gräbedünkel AT09

Manuel Gutekunst AT09

Datum: 30.04.2011

Projektbetreuer: OStR Dipl.-Ing. Joachim Lindner

OStR Dipl.-Ing. Christoph Kronenburg

Erklärung

Wir versichern, dass wir diese Facharbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt haben. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

30.04.2011

.....

Datum Unterschrift

30.04.2011

.....

Datum Unterschrift

Inhaltsverzeichnis

Vorwort

Das hier dokumentierte Projekt wurde im Rahmen einer Projektarbeit an der Fachschule des Balthasar-Neumann-Technikums in Trier durchgeführt. Als zeitlicher Rahmen für die Informationsbeschaffung, Durchführung und Dokumentation liegt ein 80 Stunden Modul zugrunde. Diese Technische Dokumentation dient zum Einen als Teil der Bewertungsgrundlage für die Projektarbeit und zum Anderen als Informationsquelle bzw. Anleitung für Interessierte, um ein solches oder ähnliches Projekt zu realisieren. Es wird in den Kapiteln Schritt für Schritt die theoretische Grundlage und die praktische Umsetzung beschrieben und es ist eine Auflistung der verwendeten Soft- und Hardware für den Nachbau der lernfähigen Fernbedienung mit webbasierender Bedienoberfläche enthalten. Die Programme, die im Zuge dieser Projektarbeit entstanden sind, stehen für die Verwendung und Weiterentwicklung frei zur Verfügung.

Durch die lernfähige Infrarotfernbedienung mit webbasierender Bedienoberfläche soll ein Teil für ein Konzept eines voll automatisierten Haushaltes entstehen. In so einem Haushalt kann jedes Gerät von überall und jederzeit bedient werden. Inhalt dieser Projektarbeit ist die Realisierung einer Soft- und Hardwareschnittstelle, über die ein Fernsehgerät des Herstellers Phillips nach dem RC5-Standard bedient werden kann. Die Bedienung kann von einem PC oder Notebook mit einem gängigen Webbrowser und Zugang zu dem Netzwerk der Steuerung erfolgen. Für die Bedienung des Fernsehgerätes ist die Sichtverbindung der Schnittstelle zu dem Fernsehgerät Voraussetzung. Ein Anwendungsbeispiel hierfür ist die Bedienung der Radiofunktion des Fernsehers (Senderwahl und Änderung der Lautstärke).

Auf Grund der Rahmenbedingungen der Projektarbeit wurde nur die Bedienung **eines** Gerätes nach dem RC5-Standard realisiert. Um weitere Unterhaltungselektroniken bedienen zu können, müssen die benötigten Kommunikation-Standards gegebenenfalls eingebunden werden. Desweiteren müssen entsprechende Möglichkeiten für die Bedienoberfläche geschaffen werden, damit mehrere Geräte verwaltet werden können. Dafür wäre dann auch die Anpassung der Kommunikation zwischen Bedienoberfläche und Steuerung notwendig. Mit einer Erweiterung wäre es dann z.B. möglich, eine Stereoanlage immer und von überall bedienen zu können. Ein Anwendungsbeispiel dafür ist die Umschaltung eines Tonsignals auf einen anderen Kanal für einen entsprechenden Raum.

1 Systemübersicht

Von einem Notebook oder PC, der Zugang zu dem Netzwerk der Steuerung hat, kann das Fernsehgerät mit Hilfe der Bedienoberfläche, welche mit jedem gängigen Webbrowser aufgerufen werden kann, bedient werden. Zunächst werden die Signale der Fernbedienung des Gerätes mit dem IR-Empfänger eingelesen und im Sendespeicher der Steuerung abgelegt. Ist dieser Vorgang abgeschlossen, können die gespeicherten Signale aus dem Sendespeicher der Steuerung aufgerufen werden und mit dem IR-Sender an das Fernsehgerät gesendet werden. Für diese beiden Funktionen sind auf der Bedienoberfläche ein Sende- und ein Programmiermodus eingerichtet, welche in dem entsprechenden Kapitel genauer beschrieben werden. In Abbildung 1 ist eine Übersicht des Systems mit den Geräten und deren Verbindung dargestellt. Natürlich kann die Bedienoberfläche auch von Teilnehmern aufgerufen werden, welche sich außerhalb des LAN befinden aber einen Zugriff darauf haben.

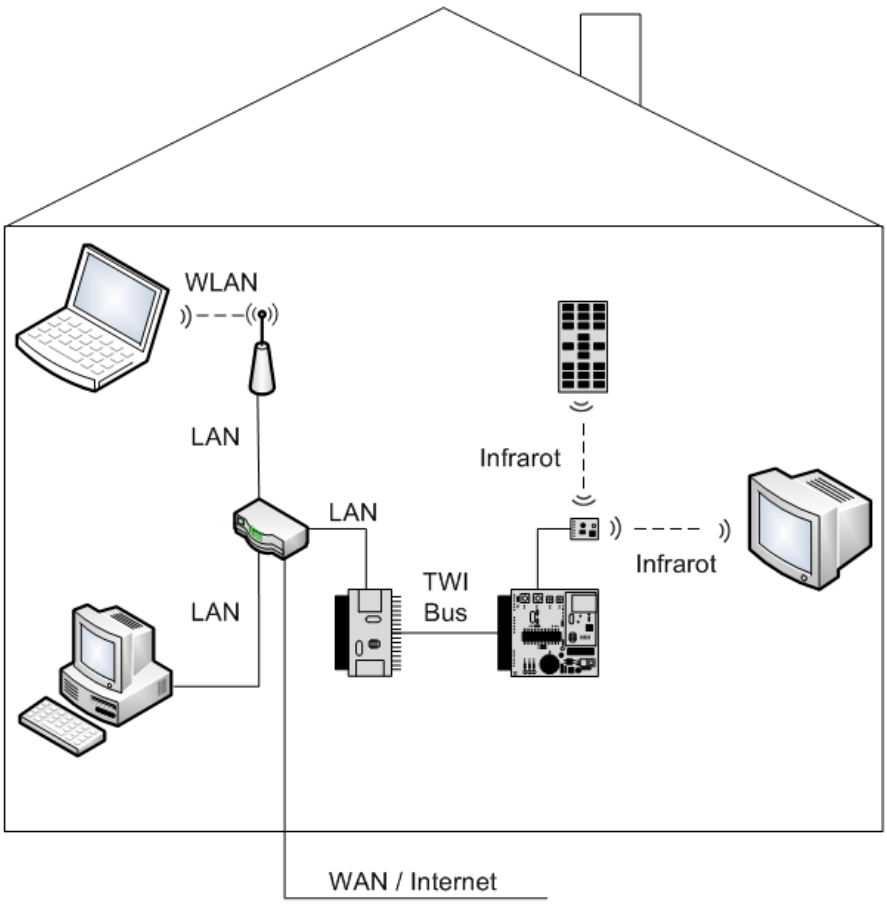


Abbildung 1 Systemübersicht

1.1 Software

Tabelle 1 Verwendete Software

Projektieroberfläche für Anwenderprogramm des Mikrocontrollers	AVR Studio 4
Projektieroberfläche zum Erstellen eines HTML Dokumentes	Evrsoft First Page 2006

1.2 Hardware

Tabelle 2 Verwendete Hardware

	Produkt	Preis
Mikrocontroller	myAVR Board MK2, bestückt http://shop.myavr.de/Systemboards/myAVR%20Board%20MK2,%20best%C3%BCckt.htm?sp=article.sp.php&artID=40	49,00 €
Ethernetboard	myEthernet http://shop.myavr.de/Add-Ons%20und%20Module/myEthernet.htm?sp=article.sp.php&artID=100065	59,00 €
Infrarot-Empfänger	IR-Empfänger-Modul Vishay TSOP1836 http://www.conrad.de/ce/de/product/171107/IR-EMPFAeNGER-MODUL-TSOP1836-4836	1,55 €
Infrarot-Sender	IR-Sende-Diode Vishay 5202 http://www.conrad.de/ce/de/product/184551/INFRAROT-SENDE-DIODE-TSUS-5202-CQW-13	0,57 €
Widerstände	1x100 Ω 1x33 Ω	
Kondensator	1x4,7 µF	

Bei der Erstellung der IR-Hardwareschnittstelle wurde die Schaltung aus dem Datenblatt des Herstellers verwendet (Abbildung 47). Als Vorwiderstand für die IR-Sendediode ergibt sich ein Widerstand von 33 Ω.

2 Infrarot

Da die meisten Geräte für Unterhaltungsmedien nur über eine IR-Schnittstelle verfügen, ist es notwendig Sichtkontakt bei der Bedienung des Gerätes zu haben. Dies kann jedoch ein Problem darstellen, wenn das Gerät von einem anderen Raum aus bedient werde soll. In diesem Kapitel wird eine Schnittstelle beschrieben, auf die der Bediener von nahezu überall zugreifen kann, wodurch es nicht mehr notwendig ist, dass das Bediengerät des Bedieners direkten Sichtkontakt zu dem zu bedienenden Gerät hat. Dies übernimmt die in diesem Kapitel beschriebene Schnittstelle.

2.1 RC5 Protokoll

Die Telegramme des RC5-Protokolls sind nach dem Manchester-Verfahren codiert. Wie in Abbildung 2 dargestellt, ist eine negative Flanke in der Bit-Mitte als eine logische 0 und eine positive Flanke als eine logische 1 zu werten. Folgen zwei identische Signalzustände, ist eine weitere Flanke in die entsprechende Richtung zwischen den Bits notwendig. Soll z.B. zweimal eine logische 1 gesendet werden, muss zwischen den Bits eine negative Flanke gesendet werden, um anschließend die, für die zweite logische 1 notwendige, positive Flanke senden zu können.

Ein Telegramm des RC5-Protokolls ist 14 Bit lang. Die Sendedauer eines Bits beträgt 1178 µs, wodurch sich eine Telegrammdauer von 14892 µs bzw. 14,892 ms ergibt. Darauf folgt eine Telegrammpausendauer von 88,889 ms, was in etwa der 3,6 fachen Sendedauer eines Telegramms entspricht.

Die Abbildung 2 stellt eine frühere Ausbaustufe des RC5-Protokolls dar. In dieser Ausbaustufe beginnt das Telegramm mit zwei Startbits gefolgt von einem Togglebit, welches zur Erkennung einer gewollten mehrfachen Sendung desselben Kommandos dient. Mit den nachfolgenden fünf Adressbits können Geräteadressen von 0-15 angesprochen werden. Die verbleibenden sechs Kommandobits ermöglichen 64 verschiedene Kommandos.

In einer neueren Ausbaustufe des RC5-Protokolls, welche in diesem Projekt verwendet wird, wurde das zweite Startbit durch ein siebtes Kommandobit ersetzt, wodurch sich die Anzahl der Kommandos auf 128 verdoppelte.

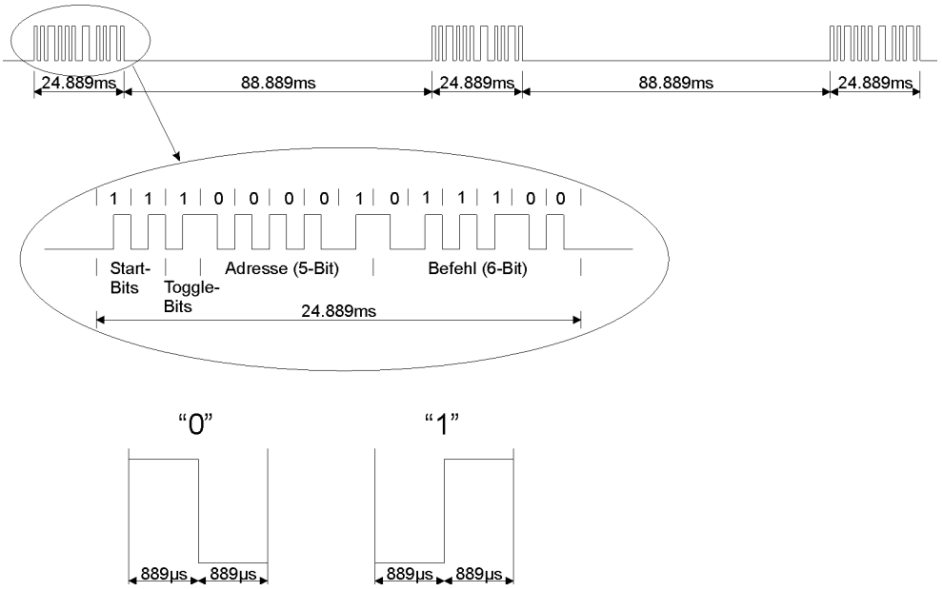


Abbildung 2 RC5-Protokollaufbau (Stefan Buchgeher 2011)

2.2 Decodierung

Wird ein Infrarotsignal nach dem RC5-Protokoll gesendet geschieht dies mit einer Trägerfrequenz von 36 kHz. Das bedeutet, dass während des IR-Pulses, wie in Abbildung 2 dargestellt, viele Impulse mit der Trägerfrequenz, gemäß Abbildung 3, gesendet werden.

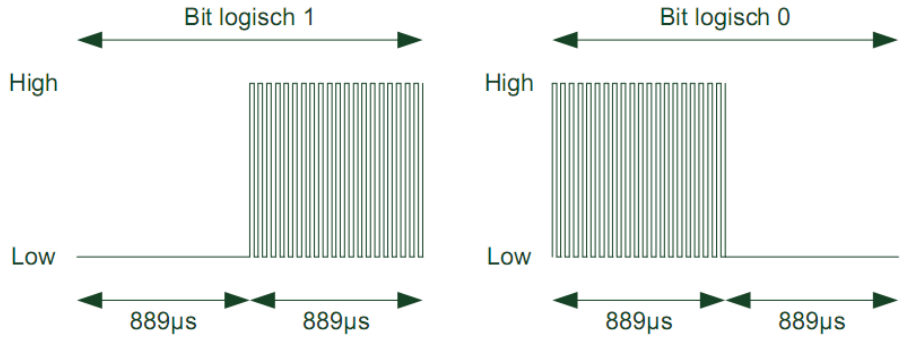


Abbildung 3 IR-Impulse der Fernbedienung

Um eine Trägerfrequenz in diesem Bereich mit dem Mikrocontroller auswerten zu können, wäre eine vielfache Taktrate des ATmeg8 notwendig. Da der verwendete Empfänger TSOP1836 die Trägerfrequenz filtert (siehe Abbildung 4), ist für die Decodierung des IR-Signals die Taktrate des ATmega8 mehr als ausreichend. Desweiteren invertiert der Empfänger das empfangene Signal, was zur Folge hat, dass eine negative Flanke einer logischen 1 entspricht und eine positiv Flanke einer logischen 0.

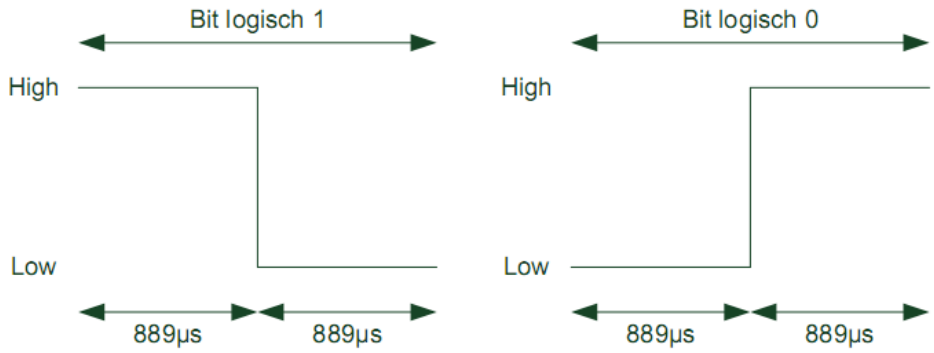


Abbildung 4 Elektrisches Signal des Empfängers

Wie das Sprichwort *„Viele Wege führen nach Rom“* schon sagt, gibt es immer mehrere Möglichkeiten ein Ziel zu erreichen. In Abbildung 5 ist eine Möglichkeit zur Decodierung des RC5-Protokolls dargestellt.

Wird eine negative Flanke erkannt (eine logische 1), weiß der Mikrocontroller, dass das Telegramm beginnt und das Startbit wird gespeichert. Nun wird ca. 1335 µs, was einer dreiviertel Periodendauer entspricht, gewartet, damit eine eventuelle Flanke zwischen den Bits nicht fälschlicherweise als eine Signal-Flanke gewertet wird. Anschließend wird für eine Zeit von ca. 889 µs auf eine Signalflanke gewartet und beim Auftreten dieser Flanke der entsprechende logische Signalzustand gespeichert. Gleichzeitig wird die Wartezeit von 889 µs neu gestartet. Nach diesem Verfahren wird das komplette Telegramm erfasst und durch das Ende des Wartens auf eine Signalflanke, bei dem keine Signalflanke auftritt, die Empfangsroutine beendet.

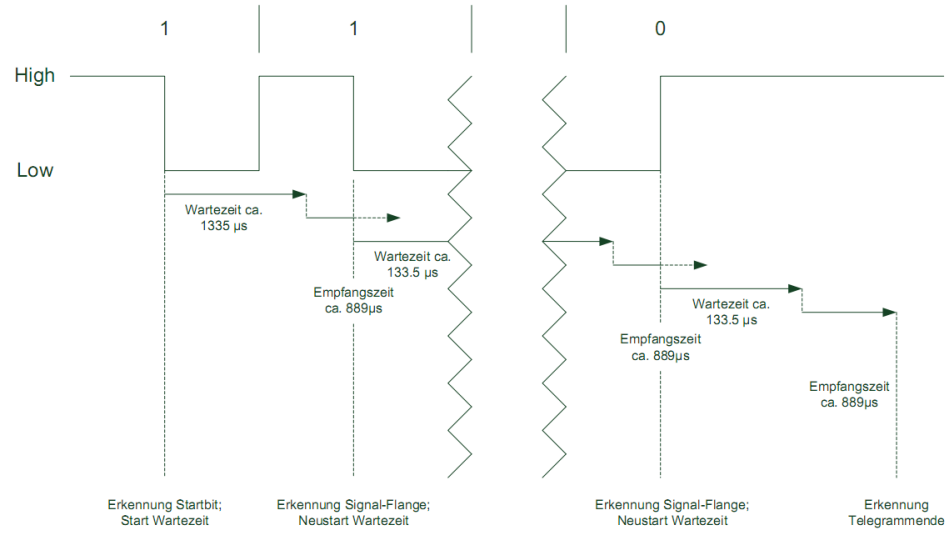


Abbildung 5 RC5-Decodierung

2.2.1 Erläuterung der Empfangsroutine

Bevor auf den Quelltext eingangen wird, folgt eine Erläuterung des beschriebenen Decodierverfahrens in Text und Grafik. Wird ein Low-Pegel an dem Eingang des IR-Empfängers festgestellt (Beginn des Telegramms), wird die Empfangsroutine gestartet.

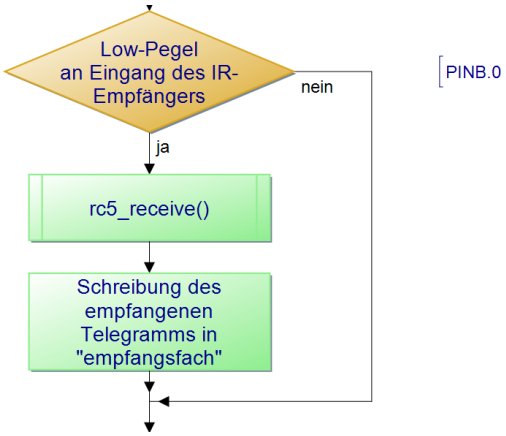


Abbildung 6 Aufruf der RC5-Empfangsroutine

Zunächst werden die für den Empfangsvorgang notwendigen Variablen deklariert und initialisiert und eine logische 1 an die Stelle des wertniedrigsten Bits in einem temporären Empfangsspeicher geschrieben. Diese logische 1 ist das Startbit (Empfangsroutine wurde ausgelöst – d.h. Startbit wurde bereits empfangen).

Nun beginnt die, mit einer Schleife realisierte Wartezeit von ca. 1335 µs, welche einer dreiviertel Periodendauer entspricht. Da nach dieser Wartezeit eine eventuelle Flanke, welche keine Signal-Flanke ist, bereits aufgetreten ist, wird der Signalzuständ am Eingang des IR-Empfängers gespeichert. Dies ist notwendig um innerhalb der nachfolgenden Wartezeit (ca. 889 µs – halbe Periodendauer) festzustellen, ob es sich um eine positive oder um eine negative Signal-Flanke handelt.

Tritt eine Pegeländerung an dem Eingang des IR-Empfängers auf ist lediglich eine anschließende Abfrage des gespeicherten Signalzustandes notwendig, um zu wissen, ob es sich um eine positive oder negative Flanke handelt. Bei einer positiven Flanke, welche einer logischen 0 entspricht, wird der Inhalt des temporären Telegrammzwischenspeichers um eine Stelle nach links geschoben. Bei einer negativen Flanke, welche einer logischen 1 entspricht, wird zusätzlich noch eine logische 1 an der Stelle des wertniedrigsten Bits eingetragen. Da nun der Empfang eines Bits abgeschlossen ist, startet der Empfang des nächten Bits mit dem erneuten Beginn der ersten Wartezeit.

Ist hingegen keine Signalflanke aufgetreten, wird dies als Telegrammende gewertet. Es folgt nun eine Abfrage bei welcher festgestellt wird, ob sich an der Stelle des Startbits im temporären Telegrammzwischenspeicher eine logische 1 befindet. Ist dies der Fall, wird das empfangene Telegramm von der Empfangsroutine zurückgegeben, ansonsten wird der Wert 1 zurückgegeben.

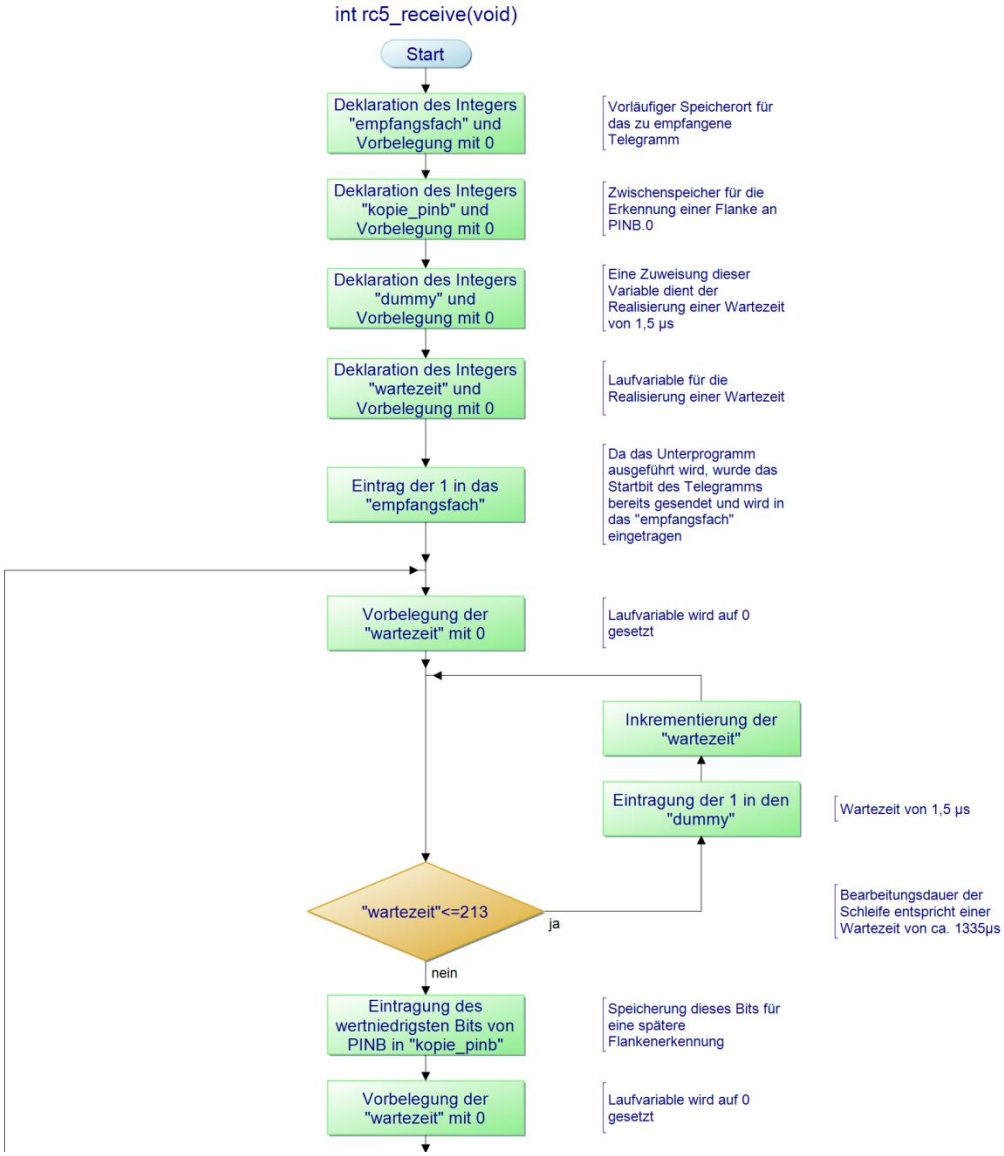


Abbildung 7 RC5-Empfangsroutine Teil1

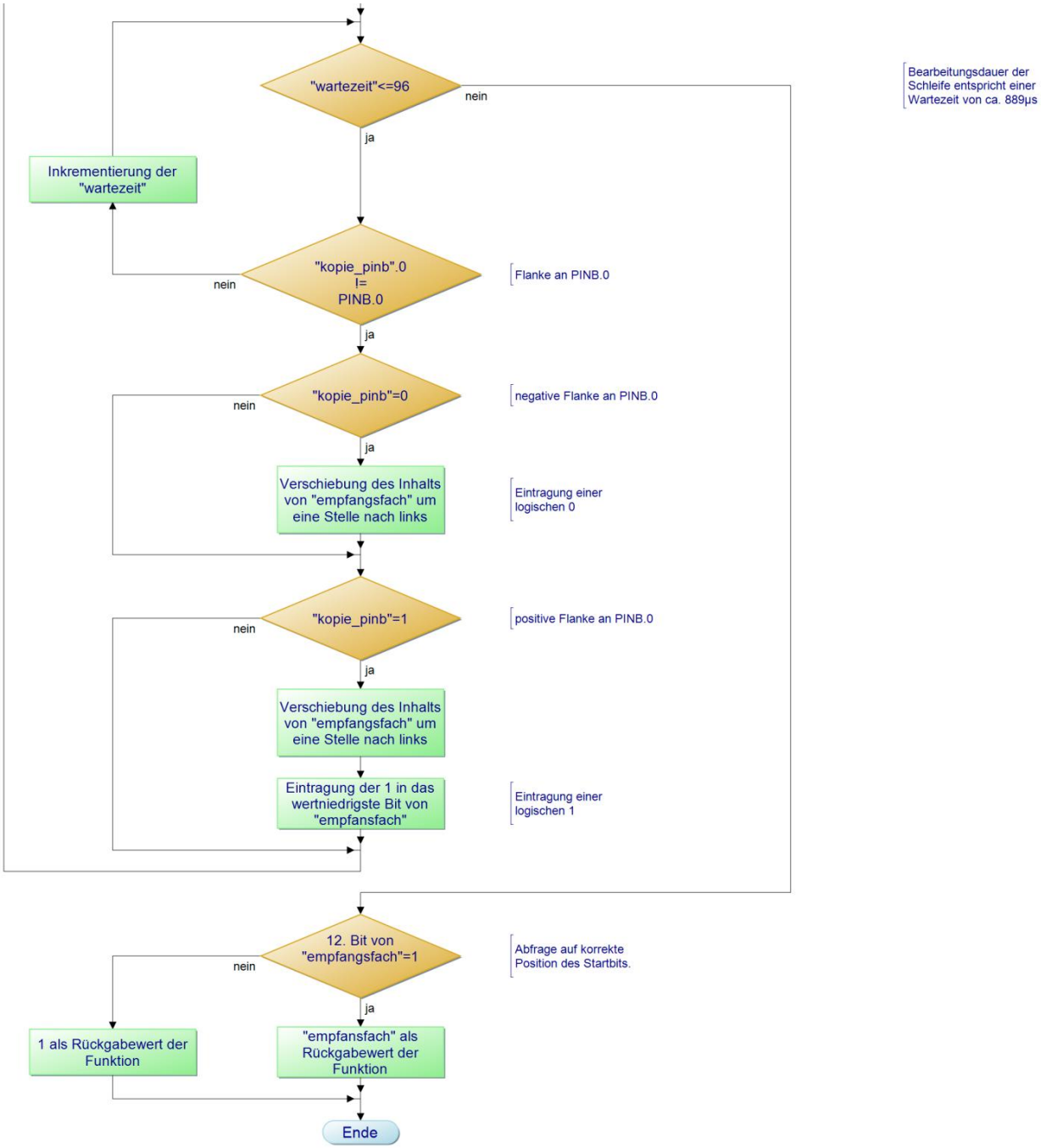


Abbildung 8 RC5-Empfangsroutine Teil2

Wie bereits erwähnt wird am Ende der Empfangsroutine das Startbit überprüft. Dadurch wird mit hoher Wahrscheinlichkeit erkannt, wenn es sich um ein falsches Telegramm handelt, welches von Störsignalen von anderen Quellen beeinflusst wurde. Die Abbildung 9 zeigt ein Störsignal, welches von einer Energiesparlampe ausgeht.

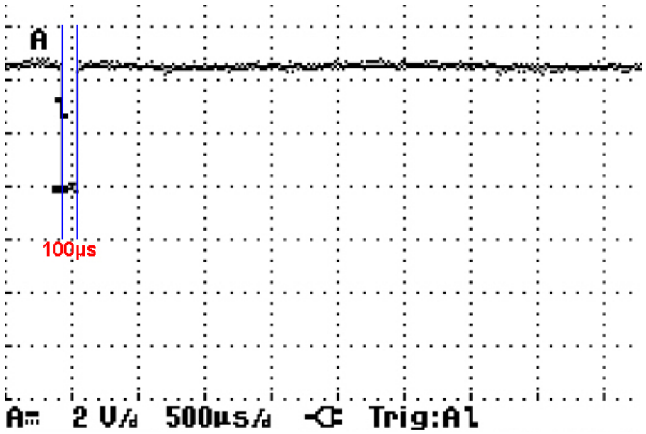


Abbildung 9 Störsignal einer Energiesparlampe

2.2.2 Quelltext der Empfangsroutine

Der Quelltext des Anwenderprogramms wurde, wie bereits erwähnt, in der Programmiersprache C verfasst. Der Aufruf der Empfangsroutine erfolgt aus dem Hauptprogramm in folgender Weise:

```
include <avr/io.h>
#include "rc5.h"
int empfangsfach=0;

/*********************************************************************
* Empfangsvorgang des rc5-Telegramms                                 *
*********************************************************************/
if (bit_is_clear(PINB,0))// Erkennung Telegrammbeginn
{
  empfangsfach=rc5_receive();
}
```

Die Empfangsroutine, ist Teil der RC5-Bibliothek, deren Entwicklung ein wesentlicher Inhalt dieser Projektarbeit war. Nachfolgend wird der Quelltext der Empfangsroutine dargestellt.

```
#define  F_CPU 3686400
#include <avr/io.h>         // Grundbibliothek
#include <stdio.h>          // Bibliothek für Sprünge
#include "rc5.h"            // prototypen

/*****************************************************************************
*Empfangen eines Rc-5 Telegramms:                                            *
*Für die Verwendung dieser Funktion im Hauptprogramm wird                    *
*die Verwendung folgender Syntax empfohlen:                                  *
*                                                                           *
*if (bit_is_clear(PINB,0)) // Erkennung Telegrammbeginn                      *
*{                                                                          *
*  variable=rc5_receive();                                                  *
*}                                                                          *
*****************************************************************************/
int rc5_receive(void)
{
  /*In diese Variable wird das empfangene RC5 Telegramm vorläufig hineinge-
  schrieben, um anschließend zu testen, ob das Telegramm fehler-
  frei empfangen wurde.*/
  int empfangsfach=0;

  /*In diese Variable wird eine Kopie des PINB Registers geschrieben,um
  eine Flanke an dem Eingang des Empfängermuduls feststellen zu können.
  (sowohl eine negativ als auch eine positive Flanke)*/
  int kopie_pinb=0;

  /*Diese Variable wird ausschließlich dafür verwendet, Zuweisungen auszu-
  führen um Wartezeiten zu realisieren.*/
  int dummy=0;

  /*Dies ist eine Zählvariable, die für die Realisierung einer Wartezeit
  und einer Pulsweitenmodulation verwendet wird.*/
  int wartezeit=0;

  empfangsfach=1;    // Eintrag des 1. Startbits in
  anfang:
  for (wartezeit = 0;wartezeit<=213;wartezeit++)
                              // ca. 3/4 Periodendauer (1335µs)
  {
    dummy=1;
  }
  kopie_pinb=PINB & 1;       // Kopie für anschl. Fl.-Erkennung
  for (wartezeit= 0;wartezeit<=96;wartezeit++)
                                          // ca. 1/2 Periodendauer (889µs)
  {
    if ((kopie_pinb^(PINB & 1))==1)
    {
      if (kopie_pinb==0)
      {
        empfangsfach=empfangsfach*2;
      }

      if (kopie_pinb==1)
      {
        empfangsfach=empfangsfach*2;
        empfangsfach|=1;
      }

      goto anfang;
    }
  }

if((empfangsfach&0x2000)==0x2000)
  {
    return (empfangsfach);
  }
  else
  {
    return 0x0001;
  }
}
```

2.3 Codierung

Um ein Telegramm an das zu bedienende Gerät zu senden, muss es gemäß dem RC5-Standard codiert werden. Wie bereits bekannt ist, wird dafür bei einer logischen 1 während der ersten Hälfte des Bits ein Low-Pegel gesendet und während der zweiten Hälfte ein High-Pegel mit der Trägerfrequenz von 36 kHz. Bei einer logischen 0 geschieht dies in umgekehrter Reihenfolge. Da auf die Erzeugung der Trägerfrequenz mit Hilfe eines externen Quarzes verzichtet wird, muss diese auch von der Steuerung erzeugt werden. Für das Senden muss nun jedes Bit des Telegramms nach der eben beschriebenen Weise codiert und gesendet werden, damit das zu bedienende Gerät die gewünschte Funktion ausführt. Um zu erkennen, dass ein Telegramm gewollt zweimal gesendet wird, muss das Togglebit beim erneuten Senden seinen Zustand wechseln.

2.3.1 Erläuterung der Senderoutine

Wird ein Telegramm in das Sendefach der Steuerung geschrieben, löst dies den Aufruf der Senderoutine aus der RC5-Bibliothek aus. Nachdem das Telegramm gesendet ist, wird das Sendefach gelöscht, um ein erneutes Senden des Telegramms zu verhindern. Sollte dies jedoch gewünscht sein, muss das Telegramm erneut in das Sendefach geschrieben werden.

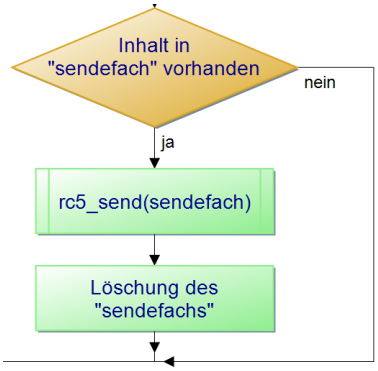


Abbildung 10 Aufruf der RC5-Senderoutine

Auch bei der Senderoutine werden zunächst die benötigten Variablen deklariert und initialisiert. Eine Ausnahme bildet die Variable, die den Zustand des Togglebits beinhaltet, da deren Inhalt ansonsten bei einem erneuten Aufruf der Senderoutine verloren gehen würde. Diese Variable wird deshalb an dem Anfang der RC5-Bibliothek deklariert und initialisiert.

Nach den Deklarationen und Initialisierungen wird zunächst der Zustand des Togglebits in der gleichnamigen Variable abgefragt. Diese beinhaltet den Zustand des Togglebits des letzten Sendevorganges. Deshalb wird ihr invertierter Zustand bei dem aktuellen Sendevorgang in das Togglebit des Telegramms geschrieben. Anschließend wird auch der Zustand der Variable selbst invertiert. Das zu sendende Telegramm befindet sich in der Variable „data“. Diese ist vom Typ Integer, welcher 16 Bit groß ist. Da das Telegramm nur 14 Bit groß ist, bleiben zwei Bits der Variable „data“ ungenutzt, worauf später noch eingegangen wird.

Der eigentliche Sendevorgang des Telegramms wird mit Hilfe einer Pulsweitenmodulation realisiert, wofür eine FOR-Schleife verwendet wird. Ein Grenzwert trennt die beiden Hälften der Pulsweitenmodulation. Dabei wird, abhängig von dem zu sendenden Bit, während der ersten und der zweiten Hälfte, entsprechend ein Low- bzw. High-Pegel gesendet. Die Erzeugung der Trägerfrequenz für das Senden des High-Pegels erfolgt durch eine ständige Invertierung kombiniert mit einer Zuweisung der Variable „dummy“, welche den Zweck einer Wartezeit erfüllt. Dies entspricht einer Pulsweitenmodulation innerhalb der einen Hälfte einer Pulsweitenmodulation (Abbildung 3).

Ist ein Bit gesendet, wird die Variable „gesendete_bits“, welche die Anzahl der gesendeten Bits enthält inkrementiert und der Inhalt der Variable „data“, um eine Stelle nach links geschoben. Anschließend wird das nächste Bit gesendet. Das aktuell zu sendende Bit ist immer das 14. Bit der Variable „data“.

Sind alle 14 Bits des Telegramms gesendet, wird der Sendevorgang beendet. Da sich noch die beiden Bits, welche zuletzt gesendet wurden in der Variable „data“ befinden, werden diese gelöscht und die Variable „gesendete_bits“ auf null zurückgesetzt. Dies ist noch ein Überbleibsel, welches in vorherigen Entwicklungsstufen der Senderoutine benötigt wurde. In der aktuellen Version wäre dies nicht mehr notwendig, da die Variable „data“ und die Variable „gesendete_bits“ bei jedem Aufruf der Senderoutine neu deklariert und initialisiert werden.

In dem nachfolgenden Programmablaufplan ist der Ablauf der Senderoutine noch einmal genauer beschrieben.

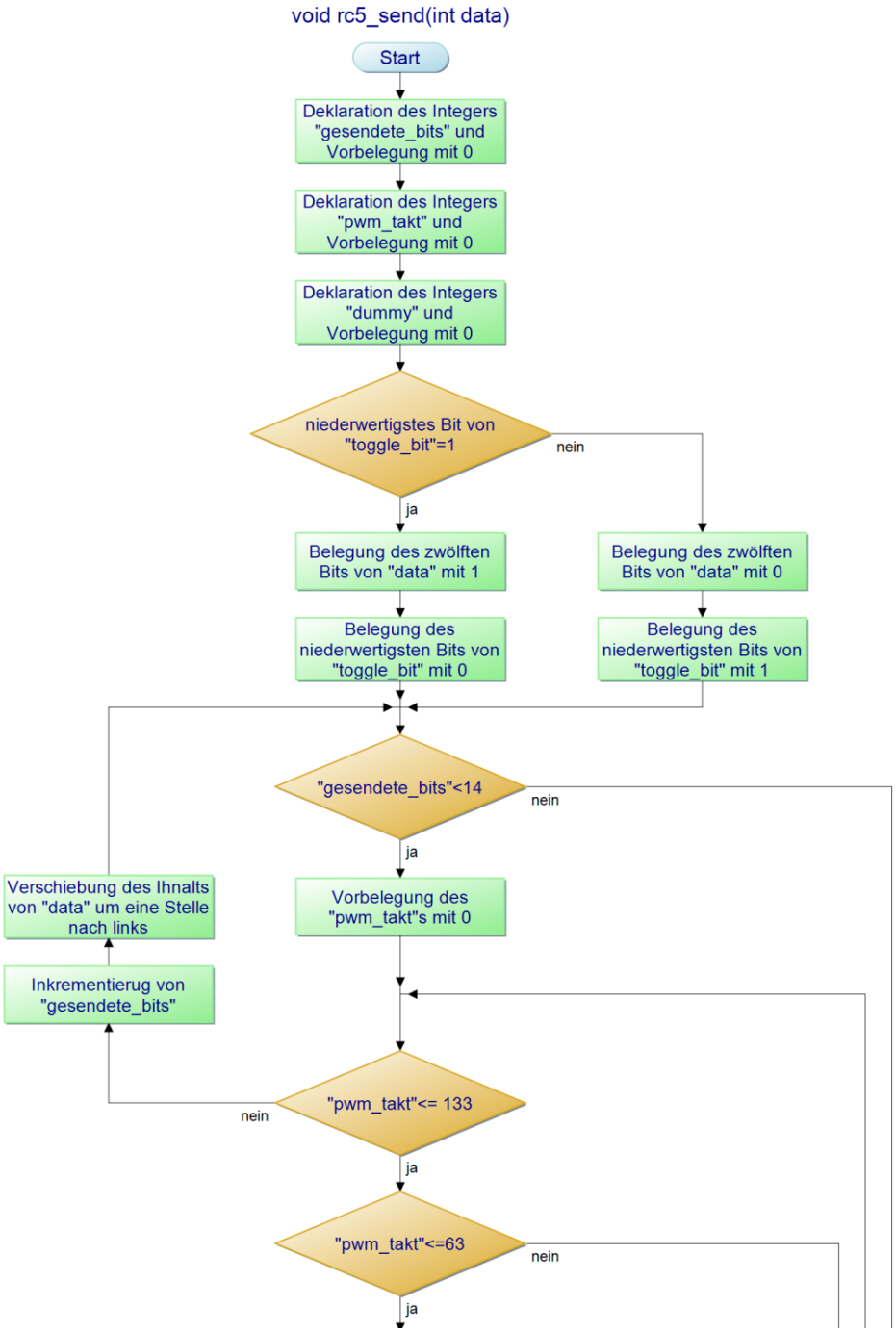


Abbildung 11 RC5-Senderoutine Teil1

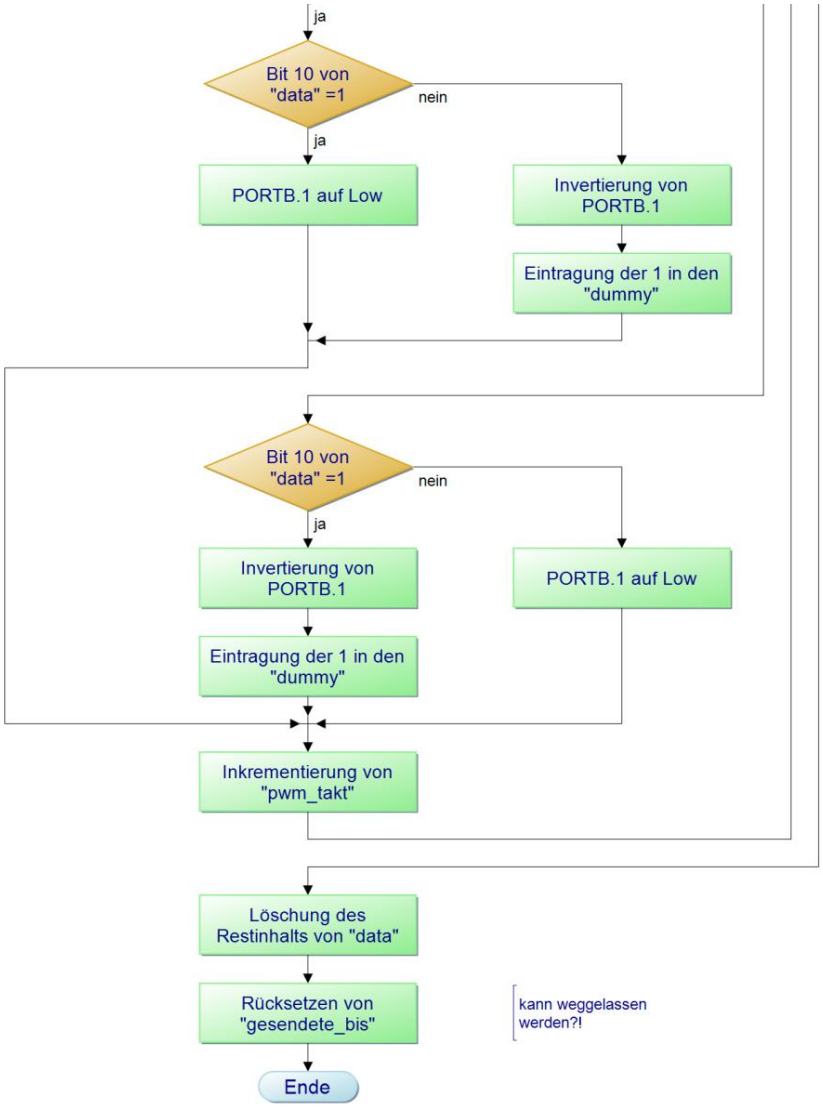


Abbildung 12 RC5-Senderoutine Teil2

2.3.2 Quelltext der Senderoutine

Der Aufruf der Senderoutine erfolgt aus dem Hauptprogramm in folgender Weise:

```
include <avr/io.h>
#include "rc5.h"

uint16_t sendefach=0;
/*********************************************************************
* Sendevorgang des rc5-Telegramms                                    *
*********************************************************************/
if(sendefach>0) // Abfrage auf Inhalt in dem sendefach
{
  rc5_send(sendefach);
  /*Da das RC5-Telegramm 14 Bit hat, das Sendefach jedoch 16, muss dieses
  nach dem Sendevorgang gelöscht werden, damit nicht ein erneuter sendevor-
  gang ausgelöst wird.*/

  sendefach=0;
}
```

Die nachfolgend dargestellte Senderoutine befindet sich wie die Empfangsroutine ebenfalls in der RC5-Bibliothek.

```
#define  F_CPU 3686400
#include <avr/io.h>        // Grundbibliothek
#include "rc5.h"           // prototypen
int toggle_bit=0;
/*Für die Realisierung einer Toggle_Funktin wird diese Variable benötigt,
die nicht in der Funktion "void rc5_send(int data)" deklariert und ini-
tialisiert werden, da sonst beim Aufruf der Funktion immer die 0 darin
stehen und somit nicht getoggelt werden würde.*/

/*************************************************************************
*Senden eines Rc5 Telegramms                                             *
*Diese Funktion gibt das zu sendende Telegramm an PortB.1 aus            *
*                                                                        *
*ACHTUNG!!!! FUNKTIONIERT NUR MIT DEM ATMEGA8!!!                         *
*Da alle Zeiten in der Senderoutine auf des Taktfrequenz des ATmega8 ba- *
*sieren und keine fertigen Wartezeiten verwendet werden, sind Trägerfre- *
*quenz und Periodendauer auch nur mit dem ATmega8 (evtl. anderer Prozessor*
*mit gleicher Taktzeit) entsprechend dem RC5-Protokoll.                  *
*                                                                        *
*Für die Verwendung dieser Funktion im Hauptprogramm wird                *
*die Verwendung folgender Syntax empfohlen:                              *
*                                                                        *
*if(data>0) // Abfrage auf Inhalt in dem data                            *
*{                                                                       *
*  rc5_dend(variable);                                                   *
*                                                                        *
*  //Da das RC5-Telegramm 14 Bit hat, das data jedoch 16, muss dieses    *
*  //nach dem Sendevorgang gelöscht werden, damit nicht ein erneuter     *
*  //sendevorgang ausgelöst wird.                                        *
*  variable=0;                                                           *
*}                                                                       *
*************************************************************************/
void rc5_send(int data)
{

  int gesendete_bits=0;
  int pwm_takt=0;
  int dummy=0;

  if ((toggle_bit&0x0001)==0x0001)
  /*Vergleich auf logisch 1 von dem toggle_bit*/
  {
```

```
  data|=0x0800;
  /*Ist der Wert des toggle_bit 's logisch 1, wird das entsprechnede
  Bit im Telegramm auf logisch 1 gesetzt.*/
  toggle_bit=0x0000;
}
else
{
  data&=0xF7FF;
  /*Ist der Wert des toggle_bit 's logisch 0, wird das entsprechnede
  Bit im Telegramm auf logisch 0 gesetzt.*/
  toggle_bit=0x0001;
}
while (gesendete_bits<14)
{
/* Solange nicht alle 14 Bits des Telegramms gesendet sind, wird
diese Schleife ausgeführt*/
  for(pwm_takt=0;pwm_takt<=133;pwm_takt++)
  {
  /*Über diese Schleife wird die Impus- bzw. Pausendauer der Bits im
  Telegramm realisiert. Der pwm_takt wird nachfolgend verwendet, um
  nachfolgende Pulsweitenmodulation zu realisieren*/
    if (pwm_takt<=63)
    {
    /*Solange der pwm_takt kleiner oder gleich 62 ist, wird die erste
    Hälfte der Pulsdauer realisiert*/
      if ((data&0x2000)==0x2000)
      /*Abfrage, ob das momentan zu sendete Bit den Wert Logisch 1
      besitzt*/
      {
        PORTB &=~(1<<1);
        /*Im Falle einer logischen 1, wird dem Ausgang für die Sende-
        diode in der ersten Hälfte der Bitdauer eine logische 0 zuge-
        wiesen*/
      }
      else
      {
        PORTB ^= (1<<1);
        /*Im Falle einer logischen 0, wird dem Ausgang für die Sende-
        diode in der ersten Hälfte der Bitdauer eine logische 1 zuge-
        wiesen*/
        dummy=0;
        /*Diese Zuweisung hat ausschließlich denn Sinn, Zeit in An-
        spruch zu nehmen, um die Abweichung der Trägerfrequenz zu
        minimieren*/
      }
    }
    else
    {
    /*Sobald der pwm_takt größer als 63 ist, wird die zweite Hälfte
    der Pulsdauer realisiert. Hirbei entsteht eine positive bzw. ne-
    gative Flange in der Hälfte, welche auf Grund der Kodierung nach
    dem Manchester-Verfahren benötigt wird. Eine positive Falnage
    entspricht einer logischen 1 und eine negative Flanke einer lo-
    gischen 0*/
      if ((data&0x2000)==0x2000)
      /*Abfrage, ob das momentan zu sendete Bit den Wert Logisch 1
      besitzt*/
      {
        PORTB ^= (1<<1);
        /*Im Falle einer logischen 1, wird dem Ausgang für die Sende-
        diode in der zweiten Hälfte der Bitdauer eine logische 1
        zugewiesen*/
        dummy=0;
        /*Diese Zuweisung hat ausschließlich denn Sinn, Zeit in An-
        spruch zu nehmen, um die Abweichung der Trägerfrequenz zu
        minimieren*/
      }
      else
      {
        PORTB &=~(1<<1);
        /*Im Falle einer logischen 0, wird dem Ausgang für die Sende-
        diode in der zweiten Hälfte der Bitdauer eine logische 0
```

```
          zugewiesen*/
        }
      }
    }
    gesendete_bits++;
    /*Da nun ein Bit gesendet wurde, wird die Anzahl der gesendeten
    Bits um 1 addiert*/
    data=(data<<1);
    /*Der Inhalt von data wird um ein Bit nach links geschoben,wo-
    durch das Bit, welches als nächstes gesendet werden soll, an die
    entsprechende Stelle in data geschoben wird.*/
  }
  data=0;
  /*Da das RC5-Telegramm 14 Bit hat, das data jedoch 16, muss dieses
  nach dem Sendevorgang gelöscht werden, damit nicht ein erneuter sendevor-
  gang ausgelöst wird.*/
  gesendete_bits=0;                                                    //kann
weggelassen werden?!
  /*Die Variable in der die gesendeten Bits gezählt werden, wird wieder
  auf 0 zurückgesetzt*/
  //toggle_bit^=0x0001;
  /*Für den Fall, dass anschließend ein Telegramm mit dem selben Inhalt
  gesendet werden soll, wird das toggle_bit invertiert, damit der emp-
  fänger das wiederholte Senden erkennen kann*/
}
```

3 Kommunikation zwischen myEthernet und MK2

3.1 Speicherverwaltung

Da die Steuerung nicht nur ein Kommando der originalen Fernbedienung beherrschen soll, ist es notwendig die Signale der originalen Fernbedienung in einem Speicher zu verwalten. Damit die Signale auch nach einer Trennung der Steuerung von der Betriebsspannung noch vorhanden sind, darf der Telegrammspeicher kein flüchtiger Speicher sein, weswegen er in dem EEPROM der Steuerung angelegt wurde. Zwar hat ein EEPROM eine endliche Anzahl von Schreibzyklen, jedoch ist die Anzahl von 100 000 Schreibzyklen (laut Hersteller) in diesem Fall mehr als ausreichend.

Für die Realisierung des Telegrammspeichers wurde daher ein vorzeichenloses Integerfeld für 32 Einträge in dem EEPROM der Steuerung angelegt.

```
#include <avr/io.h>
#include <avr/eeprom.h>
#include <inttypes.h>

uint16_t telegrammspeicher[32] EEMEM; //Telegrammspeicher für 32 Einträge
```

Aus der Telegrammlänge (14 Bit) und der Größe eines Integers (16 Bit) lässt sich ableiten, dass bei jedem Feldeintrag zwei Bits ungenutzt bleiben, was aber nicht weiter tragisch ist. Mit Hilfe des nachfolgend erklärten Kommunikationsbyte wird der Zugriff auf den Telegrammspeicher gesteuert.

3.2 Kommunikationsbyte

Mit Hilfe des Kommunikationsbytes wird der Steuerung von der Bedienoberfläche gesagt, was sie zu tun hat. Der Steuerung wird mitgeteilt, ob sie ein eingelesenes Signal, welches sich im Empfangsfach befindet, an einer bestimmten Stelle des Telegrammspeichers ablegen soll oder ob sie ein abgelegtes Telegramm aus einer bestimmten Stelle des Telegrammspeichers auslesen und in das Sendefach schreiben soll. Der Aufbau des Kommunikationsbytes ist in Abbildung 13 dargestellt.

7	6	5	4	3	2	1	0

Abbildung 13 Kommunikationsbyte

Bit 0-5 Feldadresse des Telegrammspeichers
Bit 6 1 – Schreibe Inhalt des Empfangsfach in Telegrammspeicher
0 – Schreibe Inhalt des Telegrammspeichers in Sendefach
Bit 7 Kontrollbit (ist 1 wenn Telegramm vorhanden ist)

Für die Unterscheidung der beiden Fälle werden Bit 6 und Bit 7 abgefragt und in entsprechender Weise auf den, in Bit 0 bis Bit 5, angegebenen Speicherbereich des Telegrammspeichers zugegriffen. Anschließend wird der Inhalt des Kommunikationsbytes gelöscht, damit es nicht zu einer ungewollten Mehrfachausführung des Zugriffs kommt.

Beschreibung des Sendefachs

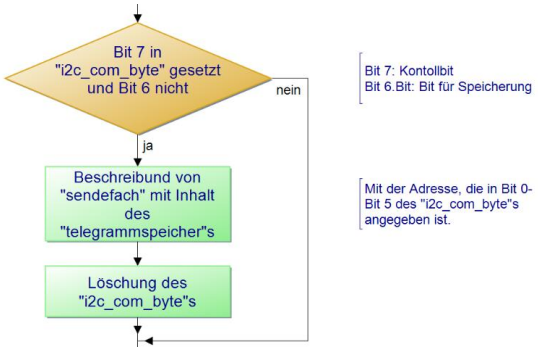


Abbildung 14 Beschreibung des „sendefach“s mit Telegramm

```
#include <avr/io.h>
#include <avr/eeprom.h>
#include <inttypes.h>
uint16_t telegrammspeicher[32] EEMEM;
char i2c_com_byte = 0;
uint16_t sendefach=0;
```

```
/*********************************************************************
* Beschreibung des „sendefachs"s mit Telegramms                      *
*********************************************************************/
if ((i2c_com_byte & 0x80)==0x80)
{
  sendefach=eeprom_read_word(&telegrammspeicher[i2c_com_byte&0x3F]);
  i2c_com_byte=0;
}
```

Beschreibung des Telegrammspeichers

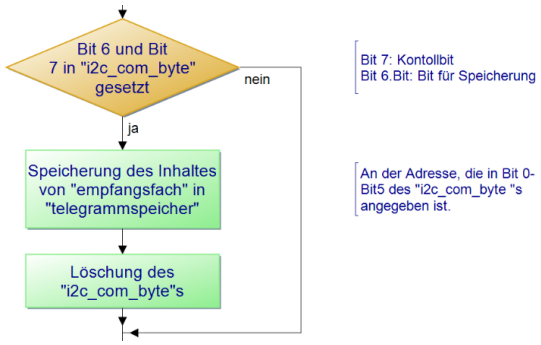


Abbildung 15 Speicherung des Telegramms in Telegrammspeicher

```
#include <avr/io.h>
#include <avr/eeprom.h>
#include <inttypes.h>
uint16_t telegrammspeicher[32] EEMEM;
char i2c_com_byte = 0;
int empfangsfach=0;

/*********************************************************************
* Speicherung des Telegramms in Telegrammspeicher                    *
*********************************************************************/
if ((i2c_com_byte & 0xC0)==0xC0)
{
  eeprom_write_word(&telegrammspeicher[i2c_com_byte&0x3F],empfangsfach);
  i2c_com_byte=0;
}
```

3.3 TWI

Über den TWI Bus, welcher auch häufig unter der Bezeichnung I^2C zu finden ist, Kommuniziert das myAVR MK2 Systemboard mit dem myEthernet Board bzw. die Bedienoberfläche mit der Steuerung.

Der TWI Bus kann als Master-Slave-Bus oder auch als Multimater-Bus betrieben werden. Die Busteilnehmer kommunizieren mit zwei Signalleitungen, über die die 8 Bit großen Dateneinheiten gesendet werden, mit einer Taktrate von 100kHz – 3,4 MHz. Der TWI Bus findet seine Anwendung häufig in der Mikrocontrollertechnik bei Peripheriegeräten, die nicht schnell sein müssen.

(wikipedia.de 2011)

3.3.1 Zugriff auf den SRAM des myEthernetboards

Da das myEthernet Board nicht die Rolle des Masters einnehmen kann (zumindest nicht mit der mitgelieferten Firmware oder nicht bekannt), übernimmt das myAVR MK2 Systemboard diese Rolle. Nun ist es aber so, dass die Steuerung bekanntlich von der Bedienoberfläche bedient wird und nicht umgekehrt. Daher wird mit der Bedienoberfläche das Kommunikationsbyte in einem vereinbarten Speicherbereich des Shared Ram des myEthernet Boards geschrieben, der regelmäßig von dem myAVR MK2 Systemboard ausgelesen und anschließend gelöscht wird.

Es ist sehr viel einfacher, wenn für die Kommunikation über den TWI-Bus eine fertige Bibliothek verwendet wird. Bei diesem Projekt kam die Bibliothek von Peter Fleury zum Einsatz, welche unter http://www.jump.to/fleury runtergeladen werden kann. In der technischen Beschreibung des myEthernet Boards kann genau nachgelesen werden, wie der Zugriff auf den Shared Ram stattzufinden hat.

Zunächst muss das myAVR MK2 Systemboard als Master in dem Bus-Netz konfiguriert werden, was nur einmal zu Beginn geschehen muss.

```
#include "i2cmaster.h"
i2c_master_init();
```

Lesender Zugriff auf das myEthernet Board:

Übermittlung der Startkondition ➢ Übermittlung der Slave-Adresse + Übermittlung der Schreibkondition ➢ Übermittlung der Adresse im Shared Ram ➢ Übermittlung der Startkondition ➢ Übermittlung der Slave-Adresse + Übermittlung der Lesekondition ➢ Lesevorgang des Bytes ➢ Übermittlung der Stoppkondition

Zu erwähnen ist, dass das „i2c_com_byte“ nur dann beschrieben wird, wenn die empfangenen Daten ungleich null waren.

Abbildung 16 Lesender Zugriff auf das myEthernet

```
include <avr/io.h>
#define  MyEthernet 0xB0
#include "i2cmaster.h"

void i2c_receive()
{

  i2c_start(MyEthernet+I2C_WRITE);        // Start, Schreiben und Slave-Adresse

  i2c_write(0x00);                        // SRAM-Adresse
  i2c_rep_start(MyEthernet+I2C_READ);     // widerholter Start, Lesen und
                                          // und Slave-Adresse
  i2c_com_byte_tmp = i2c_readNak();       // Lesevorgang aus dem SRAM

  i2c_stop();                             // Stoppkondition

  if (i2c_com_byte_tmp!=0)                // Speicherung nur bei Inhalt
  {
    i2c_com_byte=i2c_com_byte_tmp;
  }
}
```

Schreibender Zugriff auf das myEthernet Board:

Übermittlung der Startkondition ➢ Übermittlung der Slave-Adresse + Übermittlung der Schreibkondition ➢ Übermittlung der Adresse im Shared Ram ➢ Übermittlung des zu schreibenden Inhalts ➢ Übermittlung der Stoppkondition

Wie bereits erwähnt, wird bei dem schreibenden Zugriff auf das myEthernet der vereinbarte Speicherbereich im Shared Ram gelöscht, damit es nicht zu einer ungewollten Mehrfachausführung des Befehls kommt.

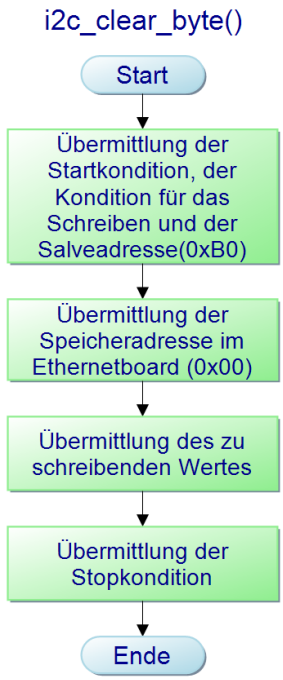


Abbildung 17 Schreibender Zugriff auf das myEthernet

```
include <avr/io.h>
#include "i2cmaster.h"
#define  MyEthernet 0xB0

void i2c_clear_byte()
{
  i2c_start(MyEthernet+I2C_WRITE);       // set device address and write mode
  i2c_write(0x00);                       // write address = 0
  i2c_write(0x00);                       // write value 0x00
  i2c_stop();                            // set stop conditon = release bus
}
```

Auslesevorgang des myEthernet Boards:

Würde der Auslesevorgang des Speicherbereichs im Shared Ram des Ethernetboards und dessen anschließende Überschreibung jeden Zyklus stattfinden, würde dies zu Problemen mit der Buskommunikation führen. Dies ist durch den zu kurzen Abstand zwischen den Vorgängen begründet. Um solche Probleme zu vermeiden, wird dieser Vorgang nur ca. alle 412,5 ms ausgeführt. Um die Zykluszeit möglichst kurz zu halten, wird der Speicherbereich im Shared Ram des Ethernetboards nur dann mit null überschrieben, wenn zuvor etwas darin stand. Wurde der Auslesevorgang ausgeführt, wird der Zähler wieder zurückgesetzt, dass die Verzugszeit von 412,5 ms von neuem beginnt.

Der nachfolgend dargestellte Auslesevorgang des Shared Rams des myEthernet Boards beinhaltet die eben erläuterten Zugriffe (lesender Zugriff – i2c_receive(); schreibender Zugriff – i2c_clear_byte();).

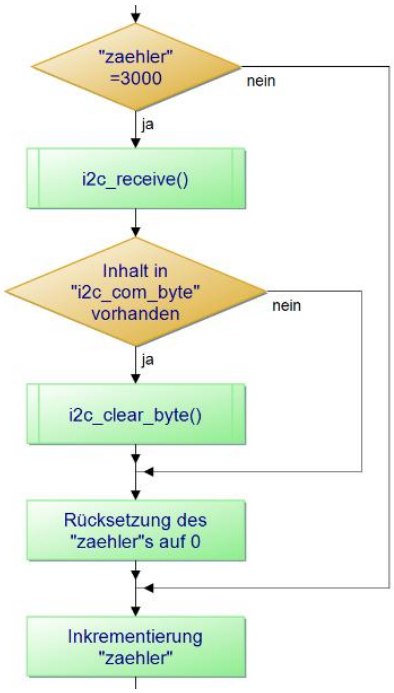


Abbildung 18 Auslesevorgang des SRAM des myEthernet Boards

```
include <avr/io.h>
#include "i2cmaster.h"
#define  MyEthernet 0xB0
int zaehler=0;
char i2c_com_byte = 0;
/*****************************************************************
*Auslesung des Bytes im MyAVR Ethernetboard                      *
*****************************************************************/
if (zaehler==30000)    // Inhalt wird ca. alle 412,5ms ausgeführt
{
  i2c_receive();       // Speicherbereich im Ethernetboard wird ausgelesen

  if (i2c_com_byte!=0)
  {
```

```
    i2c_clear_byte(); // Speicherbereich im Ethernetboard wird mit null
                      // überschrieben (nur wenn sich ein Inhalt darin befand)
  }

  zaehler=0;
}
zaehler++;
```

4 Bedienoberfläche

Die Bedienoberfläche wird mit Hilfe einer Webseite realisiert die über einen Standard Webbrowser z.B. Mozilla Firefox aufgerufen werden kann. Diese Webseite befindet sich auf der MicroSD Karte des myEthernet. Die Homepage besteht aus drei Seiten die angewählt werden können. Home, TV und Impressum.

Ist die Seite „TV“ angewählt, wird die Webfernbedienung sichtbar mit welcher das TV Gerät bedient werden kann. Über den Button „Moduswechsel“ kann zwischen Sende- und Programmiermodus gewechselt werden. Damit der Anwender zu jeder Zeit weiß, in welchem Modus er sich befindet, wird dies über eine Grafik links neben der Webfernbedienung realisiert. Diese Grafik zeigt den jeweiligen Modus an. Im Programmiermodus kann ein Kommando der realen Fernbedienung eingelesen und auf einer Taste der Webfernbedienung gespeichert werden. Dafür muss die reale Fernbedienung auf das Empfangsmodul des Mikrocontrollers gerichtet sein. Jetzt kann die zu programmierende Taste betätigt werden. Nachdem die Taste gedrückt wurde, kann nun der gewünschte Speicherort des Kommandos auf der Webfernbedienung ausgewählt werden. Durch Drücken einer beliebigen Taste auf der Webfernbedienung wird das zuvor eingelesene Kommando auf die ausgewählte Taste projektiert. Wird der „Sendemodus“ aufgerufen, werden die zuvor eingespeicherten Signale, durch drücken einer Taste auf der Webfernbedienung, an den Fernseher gesendet. Es ist z.B. ein Umschalten der Programme oder eine Einstellung der Lautstärke möglich. In Abbildung 20 Bedienoberfläche TV ist die Bedienoberfläche mit der Webfernbedienung zu sehen.

Merke: Im „Sendemodus“ können nur die Tasten verwendet werden, die zuvor Projektiert worden sind.

Die verwendeten Grafiken auf der Webseite sind Urheberrechtlich geschützt. Sie dürfen nicht vervielfältigt oder in andere Weise genutzt werden. Die Grafiken wurden auf www.fotolia.de erworben und dürfen daher vom Käufer in diesem Projekt verwendet werden.

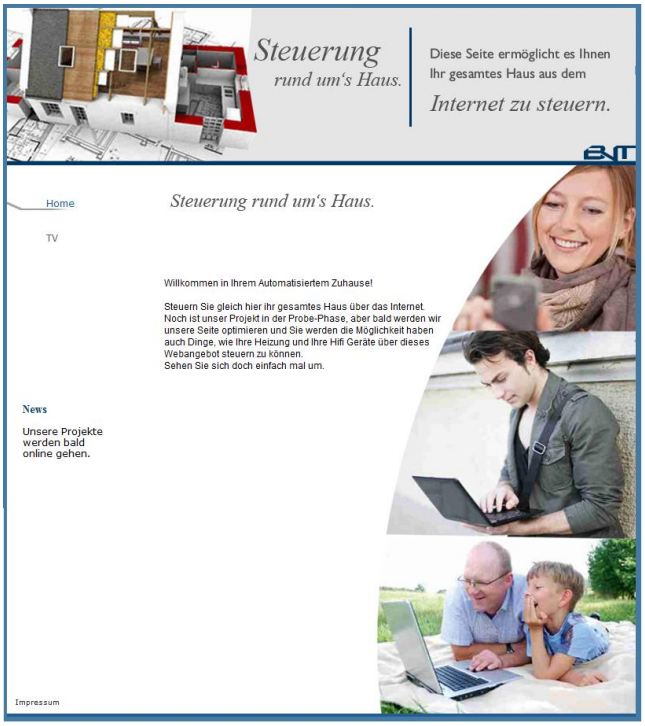


Abbildung 19 Bedienoberfläche Home

Abbildung 20 Bedienoberfläche TV

4.1 HTML & JavaScript

HTML ist eine textbasierende Auszeichnungssprache zum Strukturieren von Inhalten wie Texten, Bildern oder Hyperlinks und wird somit nicht Programmiert sondern schlicht geschrieben. Sie dient als Standardsprache im World Wide Web zum Erstellen von Webseiten. Damit das Anwendungsspektrum sich nicht nur auf Texte oder Bilder bezieht, wurden zusätzliche Programmiersprachen entwickelt die in HTML implementiert werden können. Mit diesen zusätzlichen Sprachen ist es möglich, z.B. komplexe Rechnungen oder Plausibilitätsprüfungen von Formularen durchzuführen.

Für das Ausführen von Funktionen und Anweisungen in HTML wird die Programmiersprache JavaScript verwendet. Mit dieser Programmiersprache ist es möglich FOR-Schleifen, IF-Anweisungen oder Case-Strukturen in HTML zu verwenden. Diese JavaScript Applikationen laufen im Webbrowser des Anwenders ab, während eine Webseite am Bildschirm angezeigt wird. Dadurch ist keine zusätzliche Kommunikation zum Webserver nötig. Dies spart wichtige Ressourcen der Internetverbindung und optimiert den Traffic zum Webserver. Alle Funktionen die sich im JavaScript befinden werden durch einen Verweis/Link aufgerufen der sich auf der HTML Seite befindet. Diese Art des Aufrufes ist vergleichbar mit einem Unterprogramm innerhalb eines C-Programmes. Mit Hilfe von JavaScript wurden für die Bedienoberfläche der lernfähigen Fernbedienung verschiedene Funktionen projektiert, die nun anhand von Programmablaufplänen und deren Quelltexten näher erläutert werden.

Grundkenntnisse in der Erstellung von HTML Dokumenten sind Voraussetzung, da in dieser Projektarbeit nur im begrenzten Rahmen auf grafische oder textliche Gestaltung der Webseite eingegangen wird. Eine umfassende Hilfe zum Thema HTML & JavaScript wird unter http://de.selfhtml.org/ bereitgestellt.

4.2 Moduswechsel

Damit das Kommunikationsbyte, das unter dem Punkt 3.2 behandelt wurde, richtig generiert werden kann, muss eine Unterscheidung der zwei Modi stattfinden. Dies geschieht mit Hilfe des Button „Moduswechsel“ oberhalb der Webfernbedienung und der Funktion „moduswechsel()“. Wird dieser Button betätigt, kann zwischen Programmier- und Sendemodus gewechselt werden. Durch eine dynamische Grafik links neben der Webfernbedienung wird dem Anwender angezeigt welcher Modi gerade aktiv ist. Als Standard Einstellung für die Webseite ist beim Aufruf der Seite der Sendemodus aktiv.

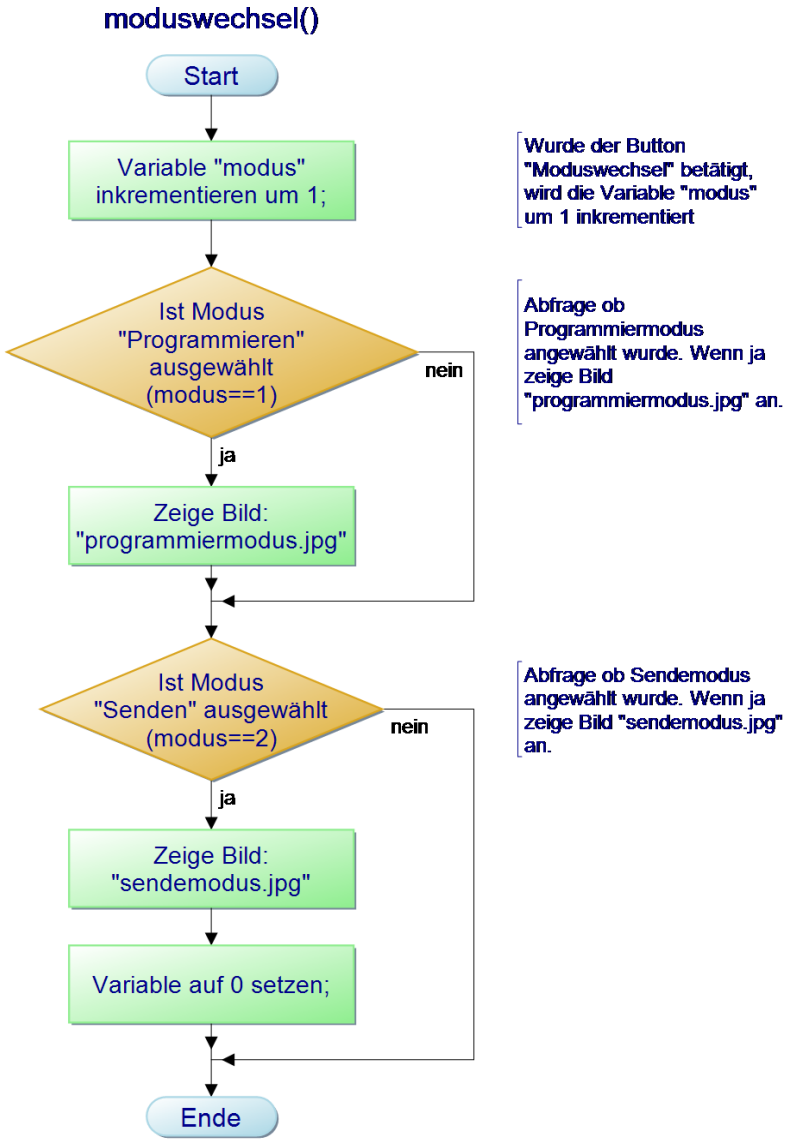


Abbildung 21 PAP Funktion moduswechsel

```
sendemodus = new Image();  //Erstelle neues Image "sendemodus"
sendemodus.src = "bilder/sendemodus.jpg";  //Zuweisung der Bilddatei

programmiermodus = new Image();  //Erstelle neues Image "programmiermodus"
programmiermodus.src = "bilder/programmiermodus.jpg";  //Zuweisung der Bilddatei


//Moduswechsel zwischen Programmier- und Sendemodus
function moduswechsel()
{
  modus = modus+1;  //Wird die Funktion aufgerufen wird "modus" um 1 inkrementiert
  if (modus==1)     //Vergleich ob "modus" == 1
  {
    document.images.modus.src = programmiermodus.src;    //Anzeige des Bildes "programmiermodus"
  }
  if (modus==2)     //Vergleich ob "modus"  = 2
  {
    document.images.modus.src = sendemodus.src;      //Anzeige des Bildes "sendemodus"
    modus=0;        //Variable "modus" den Wert 0 zuweisen
  }
}
```

Wird der Button „Moduswechsel“ betätigt, wird die oben stehende Funktion „moduswechsel()“ ausgeführt. In der Funktion wird die Variable „modus“ um eins inkrementiert, wodurch sich eine Zählung der Betätigungen ergibt. Die Variable „modus“ wird zusätzlich unter dem Punkt 4.3 verwendet, um das Kommunikationsbyte für das myAVR MK2 zu generieren. Mit den IF-Anweisungen wird nun zwischen der Anzahl der Betätigungen unterschieden um die jeweilige Grafik anzeigen zu können. Wurde die Webseite neu aufgerufen und der Button „Moduswechsel“ noch nicht betätigt, hat die Variable „modus“ den Wert 0. Geht man nach der Funktion, dürfte keine Grafik angezeigt werden. Der Webseite muss eine Startgrafik zugewiesen werden die beim ersten Aufruf angezeigt werden soll. Der folgende Quelltext zeigt die Deklaration der Startgrafik innerhalb einer HTML Datei.

```
<img src="bilder/sendemodus.jpg" alt="Sendemodus" name="modus" class="modusgrafik">
```

Zum Wechsel der Grafik wird der Tag „name“ benötigt. Mit diesem Tag ist es in JavaScript möglich, eine andere Bildquelle der zuvor deklarierten Startgrafik zuzuweisen. Hat die Variable „modus“ den Wert 1, wird der Webseite die Grafik für den Programmiermodus zugewiesen und angezeigt. Dies geschieht mit folgendem Befehl.

```
document.images.modus.src = programmiermodus.src
```

Auf dem momentan angezeigten Dokument (Webseite) wird dem Image „modus“ die Bildquelle des Programmiermodus zugewiesen.

4.3 Shared Ram des myEthernet

Das myEthernet Systemboard verfügt über 128 Byte im Shared Ram und 128 Byte im EEPROM die vom Programmierer frei verwendet werden können. Die Einstellungen über die Größe und Art des verwendeten Speichers kann über eine Konfigurationsdatei, die sich auf der MicroSD-Karte des Webservers befindet, verändert werden. Mit einer Auswahl von Befehlen können Werte in den Shared Ram bzw. das EEPROM geschrieben werden. Durch die Kommunikationsschnittstelle des TWI-Busses können Geräte, die an dem myEthernet angeschlossen sind, dessen Speicherbereiche auslesen oder beschreiben. Das Beschreiben des Shared Ram bzw. EEPROM über einen Browser wird mit Hilfe des Standard Befehls zum Ändern der reellen Pins realisiert.

```
http://IP-des-Webservers/?myChangeCmd=°o<PinNr>~<Wert>°
```

Bei Eingaben von Befehlen in die Adresszeile des Browsers bildet die IP-Nummer des Webservers den Anfang. Nun folgt die eigentliche Kommandosyntax womit der Wert eines Pins geändert werden kann, ?myChangeCmd=°o<PinNr>~<Wert>°. Da der Speicherbereich des Shared Ram kein reeller Pin ist, wird dieser als virtueller Pin bezeichnet. Für das Beschreiben des ersten Bytes im Shared Ram wird der virtuelle Pin 1000 verwendet. Die Adressen der virtuellen Pins sind in der Abbildung 22 Virtuelle Pin-Nummern des Shared Ram (projekte.myavr.de 2011) dargestellt.

Merke: Wird der Befehl innerhalb einer HTML-Datei verwendet, muss das Grad und Tildezeichen durch einen Zeichencode ersetzt werden. Für das Gradzeichen muss %B0 und für das Tildezeichen %7E eingesetzt werden, da es sonst zu Syntaxfehlern im Kommando kommt. Hierzu ein Beispiel.

```
<a href="http://IP-des-Webservers/?myChangeCmd=%B0o<PinNr>%7E<Wert>%B0" >
```

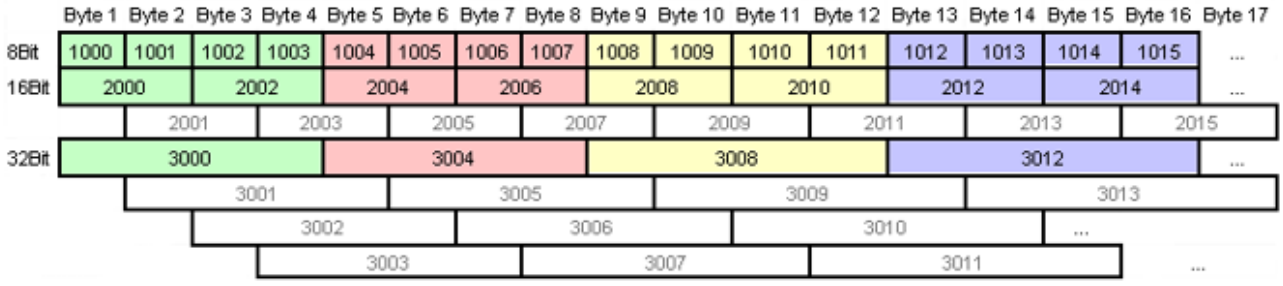


Abbildung 22 Virtuelle Pin-Nummern des Shared Ram **(projekte.myavr.de 2011)**

4.3.1 Erzeugung des Kommunikationsbytes

Damit das Kommunikationsbyte erzeugt werden kann, muss eine Funktion programmiert werden, die bei Betätigung einer Taste auf der Webfernbedienung ausgeführt wird. Beim Aufruf dieser Funktion wird als erster Schritt überprüft, in welchem Modus sich die Webfernbedienung befindet. Unter Berücksichtigung der Überprüfung wird für die beiden Modi ein unterschiedliches Kommunikationsbyte generiert. Somit ist sichergestellt, dass jede Taste in Kombination mit dem aktiven Modus ein Kommunikationsbyte generiert, das eindeutig zugeordnet werden kann.

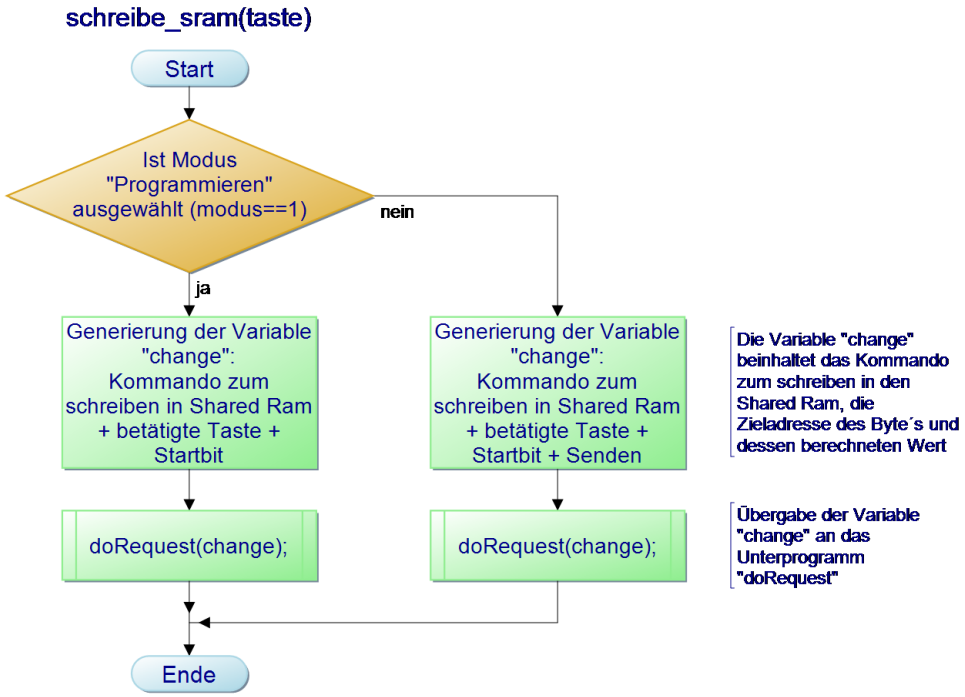


Abbildung 23 PAP Funktion schreibe_sram(taste)

Durch das Betätigen einer beliebigen Taste auf der Webfernbedienung wird die Funktion „schreibe_sram(taste)“ ausgeführt. Mit Hilfe der IF-Anweisungen innerhalb der Funktion wird der gerade aktive Modus ermittelt. Nun folgt die Zuweisung des Übergabeparameters „change“. In der Zuweisung sind die Adresse und der Befehl (?myChangeCmd=%B0o1000%7E<Wert>) als Konstanten anzusehen. Diese sind während der Bedienung nicht änderbar.

```
var kontrollbit=128;
var senden=64;

//Schreibe in SRAM des myEthernet
function schreibe_sram(taste)  //Funktion für das Schreiben in den Shared Ram
{
  if(modus==1)  //Ist Programmiermodus aktiv
  {
    var change="?myChangeCmd=%B0o1000%7E"+(taste+kontrollbit)+"%B0";  //Variable "change" wird mit
                                                                      //dem Kommando und dem Wert beschrieben
    doRequest (change);  //Unterprogramm "doRequest" für Ajax, übergabe von "change"
  }
  else  //Ist Sendemodus aktiv
  {
    var change="?myChangeCmd=%B0o1000%7E"+(taste+kontrollbit+senden)+"%B0";  //Variable "change" wird
                                                                             //mit dem Kommando und dem Wert beschrieben
    doRequest (change);  //Unterprogramm "doRequest" für Ajax, übergabe von "change"
  }
}
```

Der Wert der in das Kommunikationsbyte geschrieben werden soll, setzt sich aus drei Variablen zusammen und wird mit einer simplen Addition gebildet.

1. „taste“ Jede Taste der Webfernbedienung besitzt eine eindeutige Identifikationsnummer (0-15 sind Konfiguriert, 64 Tasten maximal im 6 Bit Adressbereich des Kommunikationsbytes) die mit dieser Variable in die Berechnung mit einfließt.
2. „kontrollbit“ Ist das Kontrollbit, welches im Kommunikationsbyte für die Erkennung eines Telegramms auf logisch 1 gesetzt werden muss.
3. „senden“ Dieses Bit dient der Unterscheidung der Modi im Kommunikationsbyte. Wird es auf logisch 1 gesetzt, ist der Sendemodus aktiv. Ist es logisch 0 befindet sich die Webfernbedienung im Programmiermodus

Wurde die Variable „change“ gebildet, wird diese an die Funktion „doRequest(fileUrl)“ übergeben. Diese Funktion ist für die Kommunikation zwischen dem Browser und dem Webserver verantwortlich.

4.4 Ajax Request

Bei traditionell arbeitenden Webanwendungen wird vom Anwender eine Anfrage an den Server geschickt, der Server wertet diese Anfrage aus und sendet dem Anwender eine neu generierte Webseite mit den angefragten Daten. Dies kann bei großen Webangeboten zu ungewollten Wartezeiten führen. Bei dem Ajax Verfahren (**A**synchronous **J**avaScript **a**nd **X**ML) handelt es sich um eine asynchrone Datenübertragung zwischen Browser und Server. Der Anwender schickt eine Anfrage an den Server und bekommt nur den Inhalt zurück, der sich geändert hat. Dadurch entstehen keine unnötigen Kommunikationsdaten zwischen Browser und Server. Für Ajax werden Browser benötigt, die mit JavaScript und XMLHttpRequest-Objekten arbeiten können. Diese zwei Standards werden heutzutage von jedem handelsüblichen Browser unterstützt.

4.4.1 Umsetzung

Die Kommunikation zwischen Browser und Webserver geschieht in der Funktion „doRequest(fileUrl)“. Beim Aufruf der Funktion wird als erster Schritt eine Variable erzeugt „req“. Dieser Variable wird kein Datentyp und kein Wert zugewiesen, da es sich um einen Rohdatentypen handelt. Mit der ersten IF-Anweisung in der Funktion wird überprüft ob ein XMLHttpRequest Objekt vom Browser erlaubt ist, dieses Objekt wird bei Browsern wie Safari, Opera, Netscape und InternetExplorer 7.0 für das Ajax Verfahren verwendet. Wird es akzeptiert, kann die Variable „req“ mit dem XMLHttpRequest Objekt beschrieben werden. Kann dieses Objekt nicht generiert werden folgt eine Kontrolle ob ein ActiveXObjekt erlaubt ist. Dieses ActiveXObjekt wird von den älteren Versionen des InternetExplorers von Microsoft verwendet. Da es zwei unterschiedliche ActiveXObjekte gibt, jeweils für InternetExplorer5 und InternetExplorer6, muss nacheinander versucht werden, diese in die Variable „req“ zu schreiben. Für den Versuch ein solches ActiveXObjekt zu erstellen gibt es in JavaScript den Befehl „try“ der über eine Fehlerbehandlung verfügt. Kann der Browser nicht mit dem erstellten Objekt umgehen, werden die Anweisungen ausgeführt, die in der Fehlerbehandlung stehen. Die Fehlerbehandlung wird mit dem Befehl „catch(e)“ aufgerufen. Kann der Browser keines der drei Objekte anlegen oder verwenden, kann nicht mit dem Ajax Verfahren gearbeitet werden.

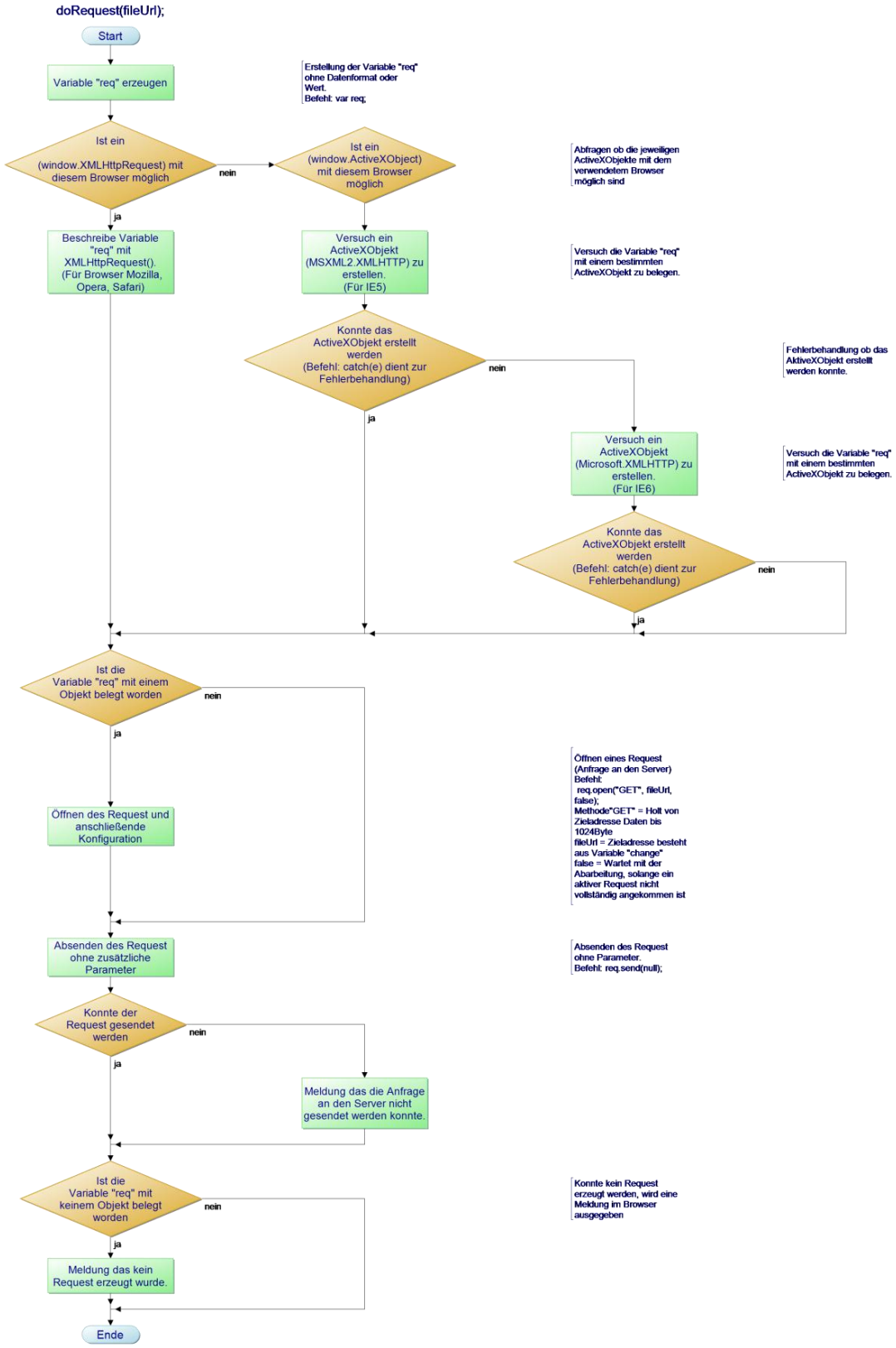


Abbildung 24 PAP Funktion doRequest(fileUrl)

Nachdem kontrolliert wurde, ob der verwendete Browser ein XMLHttpRequest Objekt in die Variable „req“ schreiben konnte, wird der Request geöffnet und konfiguriert.

```
req.open("GET", fileUrl, true);
```

Mit dem Befehl „ req.open“ wird der Request, also das erstellte XMLHttpRequest Objekt geöffnet. In den Klammern des Befehls stehen drei Parameter zur Konfiguration des Request zu Verfügung.

1. „Get“ Requestmethode mit wieviel Daten der Request Arbeiten soll. GET ist Begrenzt auf 1024 Byte. Bei größeren Datenmengen kann POST verwendet werden.
2. „fileUrl Enthält die Url an die der Request gesendet werden soll. Die Adresse setzt sich aus der Variable „change“ und der Funktion „schreibe_sram(taste)“ zusammen.
3. „true“ Ist diese Parameter auf „true“, können Anfragen asynchron an den Server gesendet werden. Bei „false“ muss die aktive Anfrage beendet sein, bevor eine neu gesendet werden kann (synchron).

Ist der Request konfiguriert, kann ein Versuch zum Senden an den Webserver unternommen werden. Die Sendung erfolgt ohne zusätzliche Übergabeparameter (null) an den Server.

```
req.send(null);
```

Konnte kein Request an den Server gesendet werden wird eine Fehlermeldung auf der Homepage ausgegeben. Bei Auftreten dieser Fehlermeldung kann ein weiteres Betätigen einer Taste auf der Webfernbedienung das Problem lösen. Wenn der Fehler immer noch besteht, sollte kontrolliert werden, ob das myEthernet Systemboard richtig angeschlossen und konfiguriert ist.

Der Quellcode des Ajax Request wurde aus der technischen Beschreibung (http://www.myavr.info/download/produkte/myethernet/techb_myethernet_de_en.pdf, techb_myethernet_de_en.pdf) des myEthernet Systemboards entnommen und um die Ausgabe der Fehlermeldungen erweitert.

```
//Ajax Request
function doRequest(fileUrl)
{                               //fileUrl = ?myChangeCmd....
  var req;                      //Variable für den Request erzeugen
  if (window.XMLHttpRequest)  //Kontrolliert ob ein XMLHttpRequest erlaubt ist
  {
    req = new XMLHttpRequest();  //erzeuge neuen Request für Mozilla,Opera,Safari
  } else if (window.ActiveXObject)  //Kontrolliert ob ein ActiveXObject erlaubt ist
  {
    try    // Versuche Einstellung für IE5
    {
      req = new ActiveXObject("MSXML2.XMLHTTP");  //erzeuge ActiveXObject für IE5
    }
    catch(e)  //Fehlerbehandlung. Wenn Browser kein ActiveXObject("MSXML2.XMLHTTP") akzeptiert
    {
      try  //Versuche Einstellung für IE6
      {
        req = new ActiveXObject("Microsoft.XMLHTTP");  //erzeuge ActiveXObject für IE6
      }
      catch(e)  //Fehlerbehandlung. Wenn Browser kein ActiveXObject("Microsoft.XMLHTTP") akzeptiert
      {
        ;
      }
    }
  }
  if (req) //Wenn Request erzeugt wurde
  {
    req.open("GET", fileUrl, true);  //Öffne Request: Methode=GET, fileUrl=Zieladresse,
                                     //true=asynchrone Datenübertragung
  }
  Try
  {
    req.send(null);  //Absenden des Request ohne Parameter
  }
  catch(e)
  {
    alert("Es konnte keine Anforderung an den Server gesendet werden!");
                                                    //Fehlermeldung
  }

  if (!req) //Konnte kein Request erzeugt werden
  {
    alert("Kann keine XMLHTTP-Instanz erzeugen!");  //Fehlermeldung
  }
}
```

5 Fazit

Da eine Motivation für die Durchführung dieser Projektarbeit darin bestand, einen Teil eines Konzeptes für einen voll automatisierten Haushalt zu entwickeln, wird nun die Erweiterung der vorhandenen Lösung beurteilt. Die nächste logische Erweiterungsstufe bestünde darin, die Kommunikationsfähigkeit der Steuerung auf weitere Infrarotübertragungsstandards zu erweitern. Um die Bedienerfreundlichkeit zu erhalten, sollte die Bedienung ohne irgendwelche Kenntnisse über den Übertragungsstandard des einzulesenden bzw. des zu sendenden Signals möglich sein. Dafür wäre es notwendig, eine Sende- bzw. Empfangsroutine zu entwickeln, die mit allen Übertragungsstandards kompatibel ist. Eine Möglichkeit dafür besteht z.B. darin, den Eingang während des Empfangs eines Signals mit Hilfe fortlaufender Interrupts abzutasten und somit eine „Aufzeichnung“ des zu empfangenden Signals zu erstellen. Da die Abtastrate bekannt wäre, könnten dadurch Rückschlüsse auf die Dauer der Pulse und der Pulspausen gezogen werden. Wichtig ist außerdem noch, das Ende des Signals zu bestimmen. Sollte dies durch das Ausbleiben einer Änderung für eine gewisse Dauer erfolgen, ist zu beachten, dass die Dauer der Pulse- bzw. Pausen von Protokoll zu Protokoll abweichen kann. Für das Senden des empfangenen Signals wäre es dann notwendig, eine Senderoutine zu entwickeln, welche die Pulse und die Pulspausen in der ermittelten Länge an den Ausgang der Steuerung gibt. Bei einer solchen Umsetzung wäre es vermutlich auch sinnvoll, für die Erzeugung der Trägerfrequenz einen externen Quarz zu verwenden, da somit der Aufwand für die Realisierung, in Anbetracht der Komplexität, wesentlich effizienter wäre.

Eine Einschränkung der Funktion besteht darin, dass nachdem die Spannungsversorgung getrennt wurde, die Signale nicht mehr korrekt an das Fernsehgerät übertragen werden. Das Gerät empfängt zwar ein Signal, jedoch reagiert es nicht. Die Ursache dafür könnte darin liegen, dass eventuell Telegrammteile verloren gehen. Bei einer Korrekturmaßnahme müsste das gesendete Signal mit einem Oszilloskop aufgezeichnet werden, wodurch dann Rückschlüsse auf den Fehler gezogen werden können.

Trotz der Einschränkungen bei der Speicherverwaltung und der Weiterentwicklung des Inhaltes dieser Projektarbeit kann ihr Ergebnis dennoch als wertvoll betrachtet werden. Die Bedienung des Fernsehgerätes des Herstellers Phillips ist unter den zu Grunde gelegten Voraussetzungen von überall möglich. Des Weiteren kann die hier entstandene Bedienoberfläche problemlos für die Bedienung weiterer Geräte erweitert werden. Das Konzept für die Kommunikation zwischen dem myEthernet und dem myAVR MK2 Systemboard kann ebenfalls diesbezüglich ausgebaut werden. Es darf außerdem nicht außer Betracht gelassen werden, dass diese Erkenntnisse über die Kriterien einer universellen IR-Schnittstelle ebenfalls ein Ergebnis dieser Projektarbeit sind, und somit ein durchaus brauchbarer Wegweiser in die geplante Richtung - die Richtung zum voll automatisierten Haushalt.

Anhang A

A.1 Programmablaufpläne

Zwar ist der Großteil der Programmablaufpläne bereits in den jeweiligen Kapiteln der technischen Dokumentation aufgeführt, jedoch wird an dieser Stelle eine zusammenhängende Übersicht geboten, welche ein besseres Verständnis des Programmablaufs ermöglichen soll.

A.1.1 myAVR MK2 Systemboard Hauptprogramm

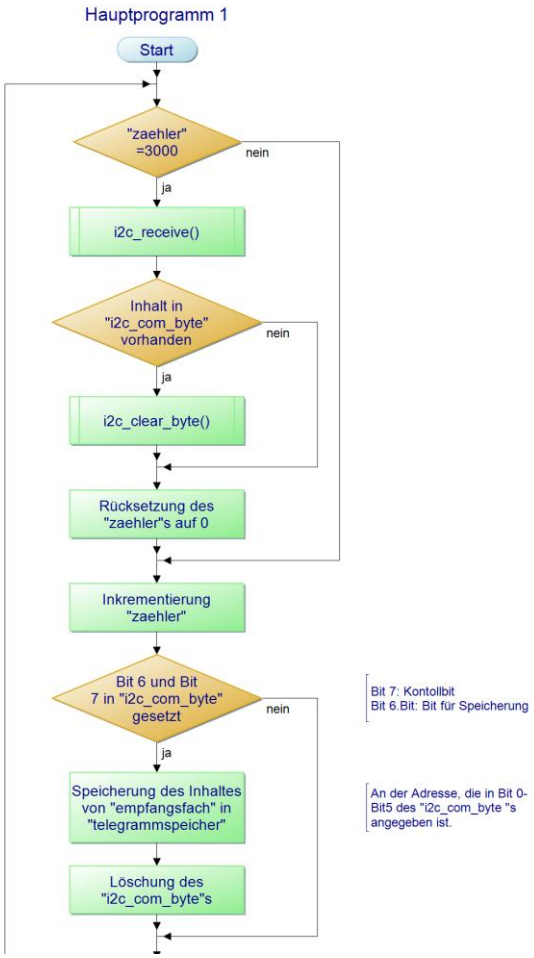


Abbildung 25 Hauptprogramm Teil1

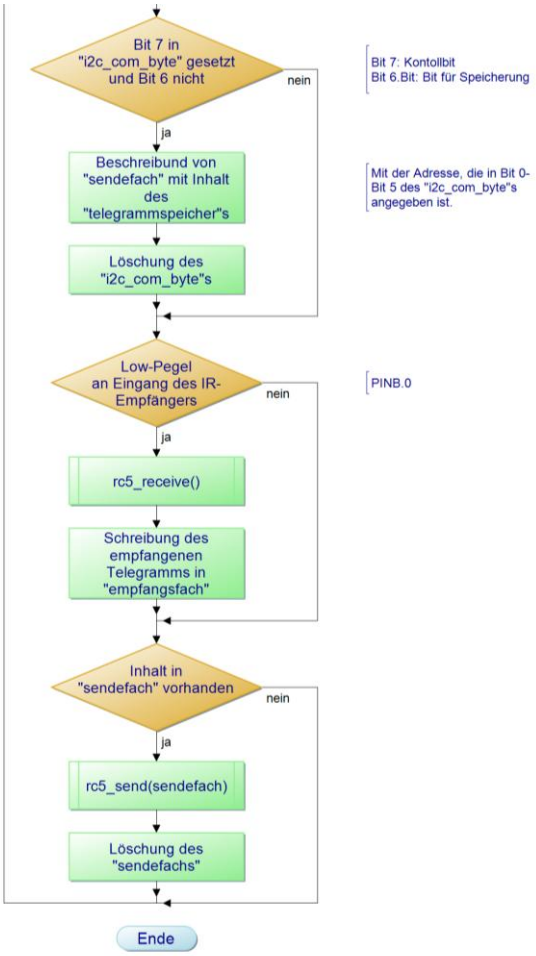


Abbildung 26 Hauptprogramm Teil2

Abbildung 27 i2c_receive

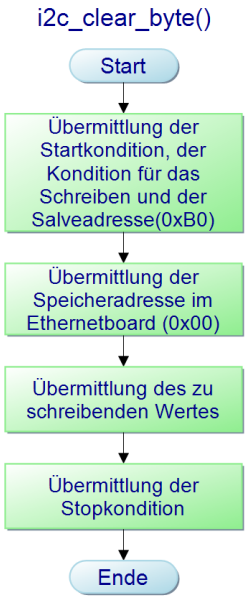


Abbildung 28 i2c_clear_byte()

A.1.2 myAVR MK2 Systemboard RC5 Bibliothek

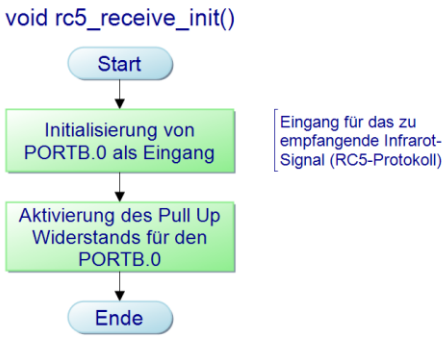


Abbildung 29 void rc5_reiceive_init()

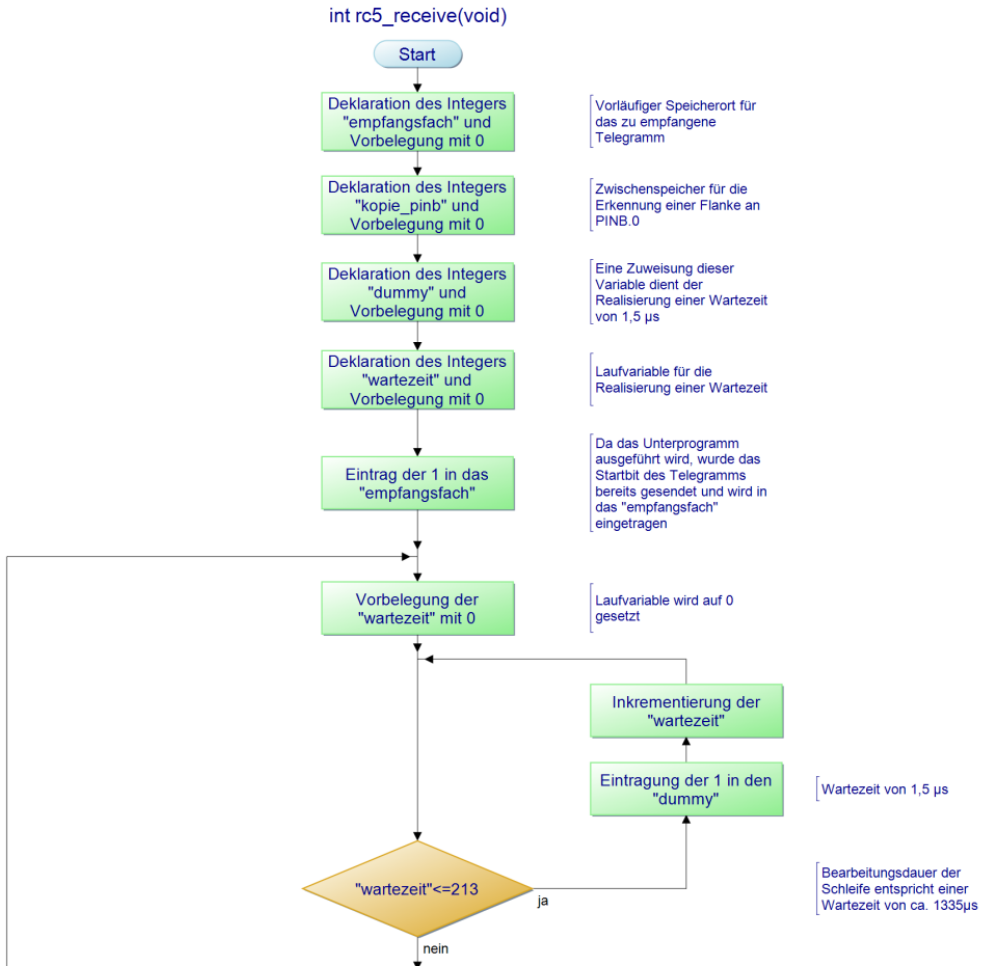


Abbildung 30 int rc5_receive (void) Teil1

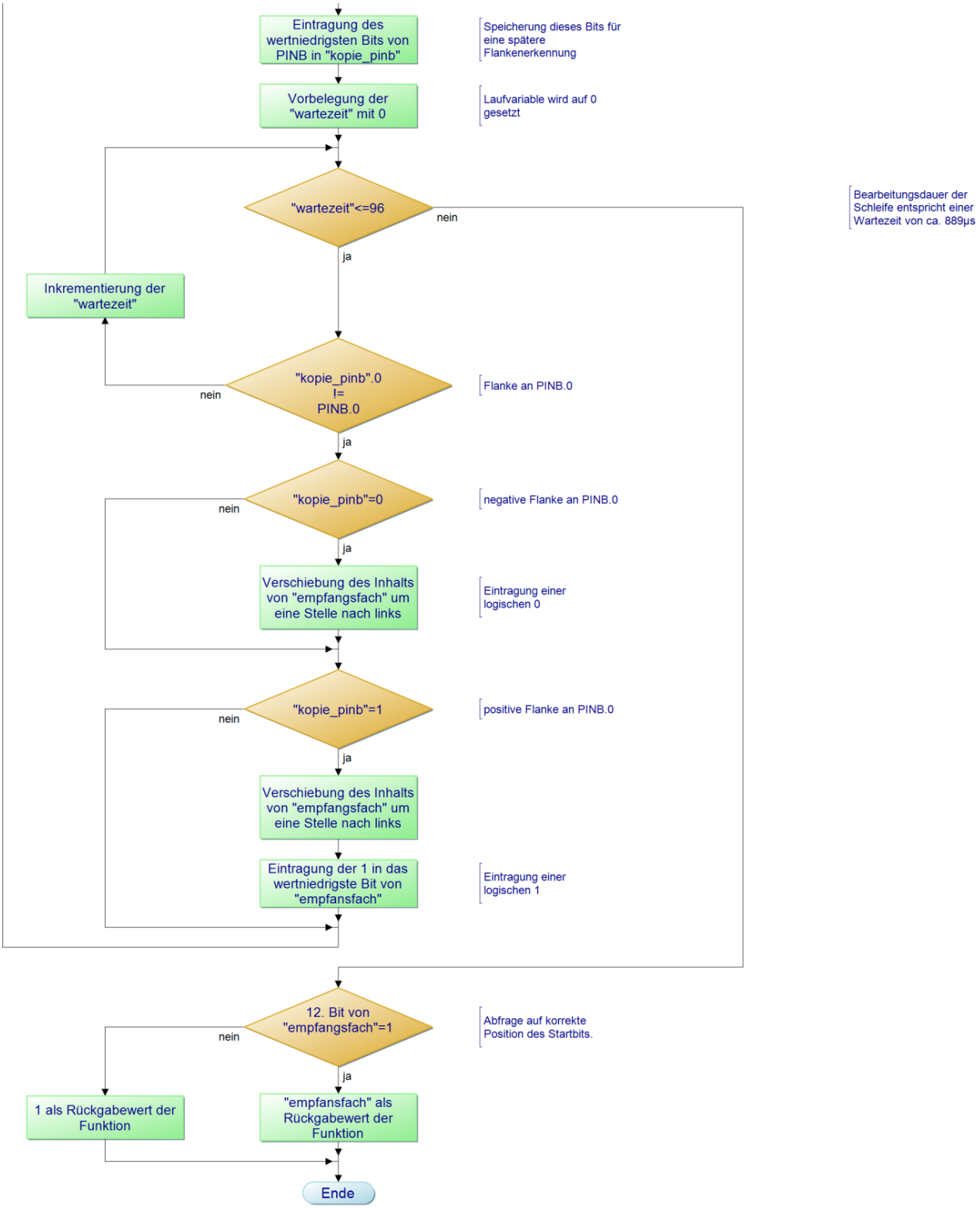


Abbildung 31 int rc5_receive (void) Teil2

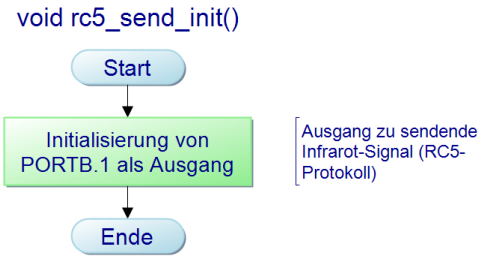


Abbildung 32 void rc5_send_init()

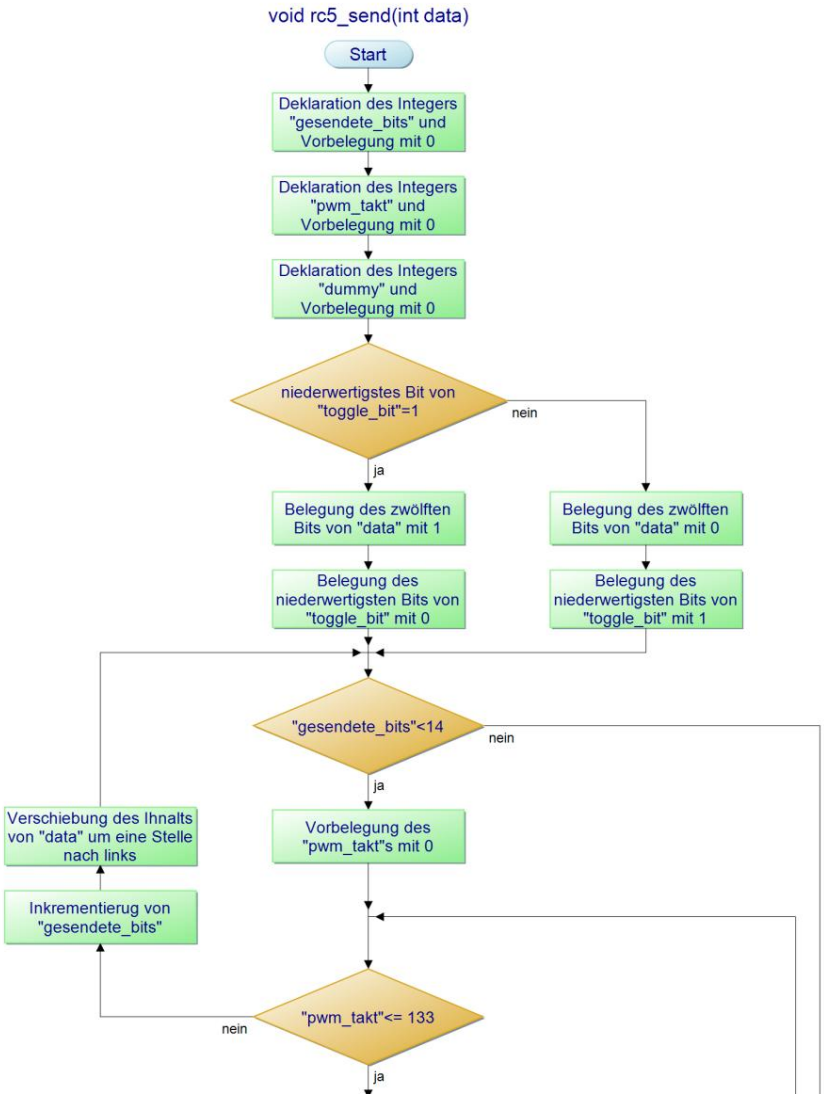


Abbildung 33 void rc5_send_(int data) Teil1

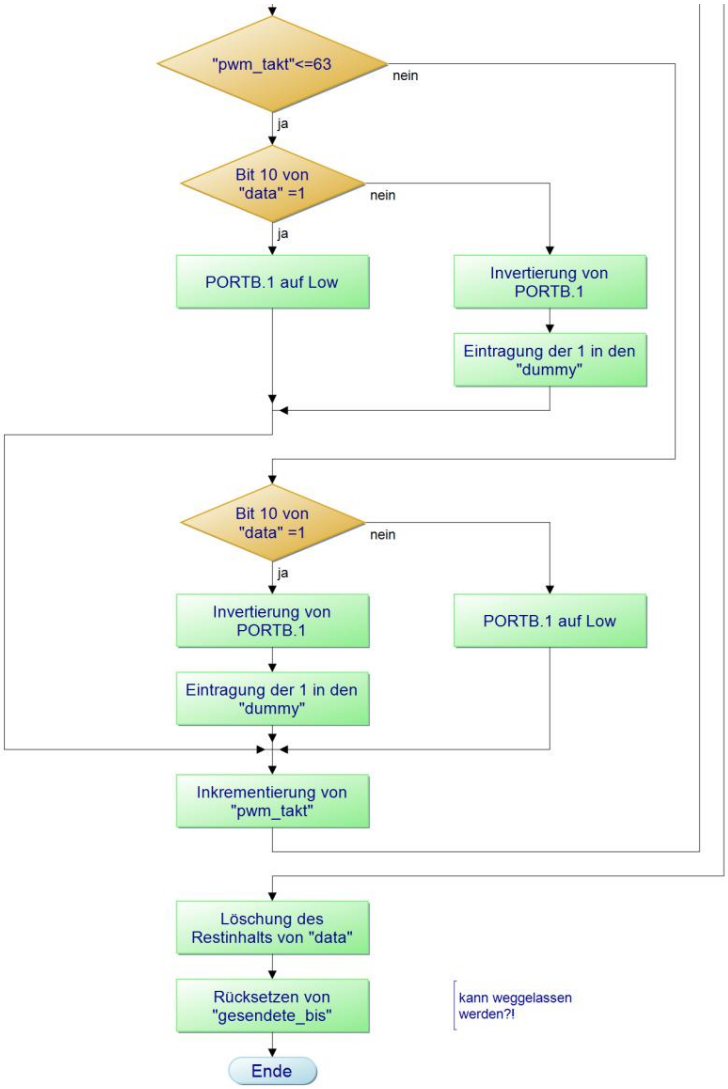


Abbildung 34 void rc5_send_(int data) Teil2

A.1.3 myEthernet Systemboard Funktion moduswechsel()

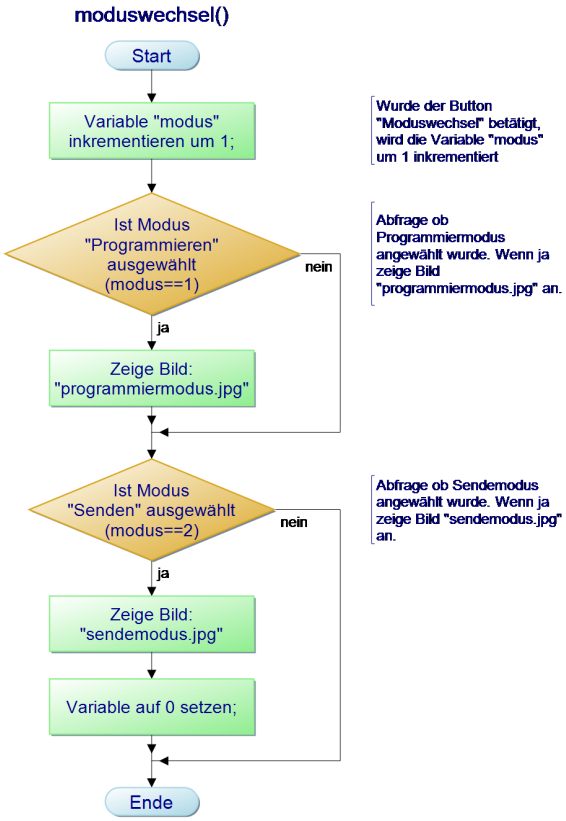


Abbildung 35 moduswechsel()

A.1.4 myEthernet Systemboard Funktion schreibe_sram(taste)

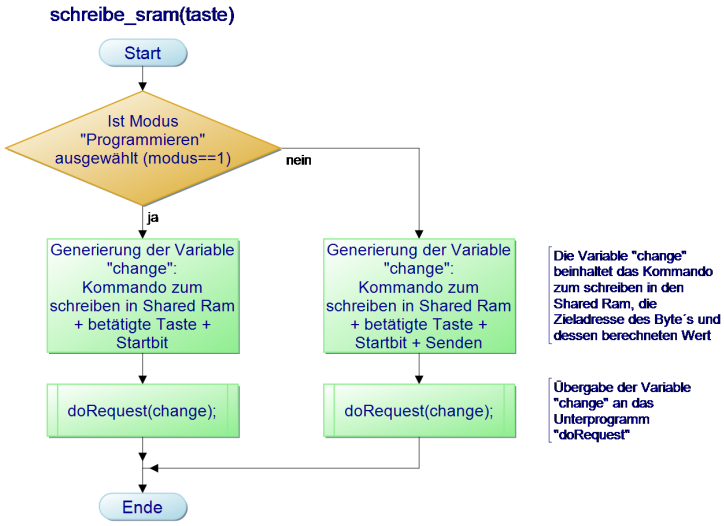


Abbildung 36 schreibe_sram(taste)

A.1.5 myEthernet Systemboard Funktion doRequest(fileUrl)

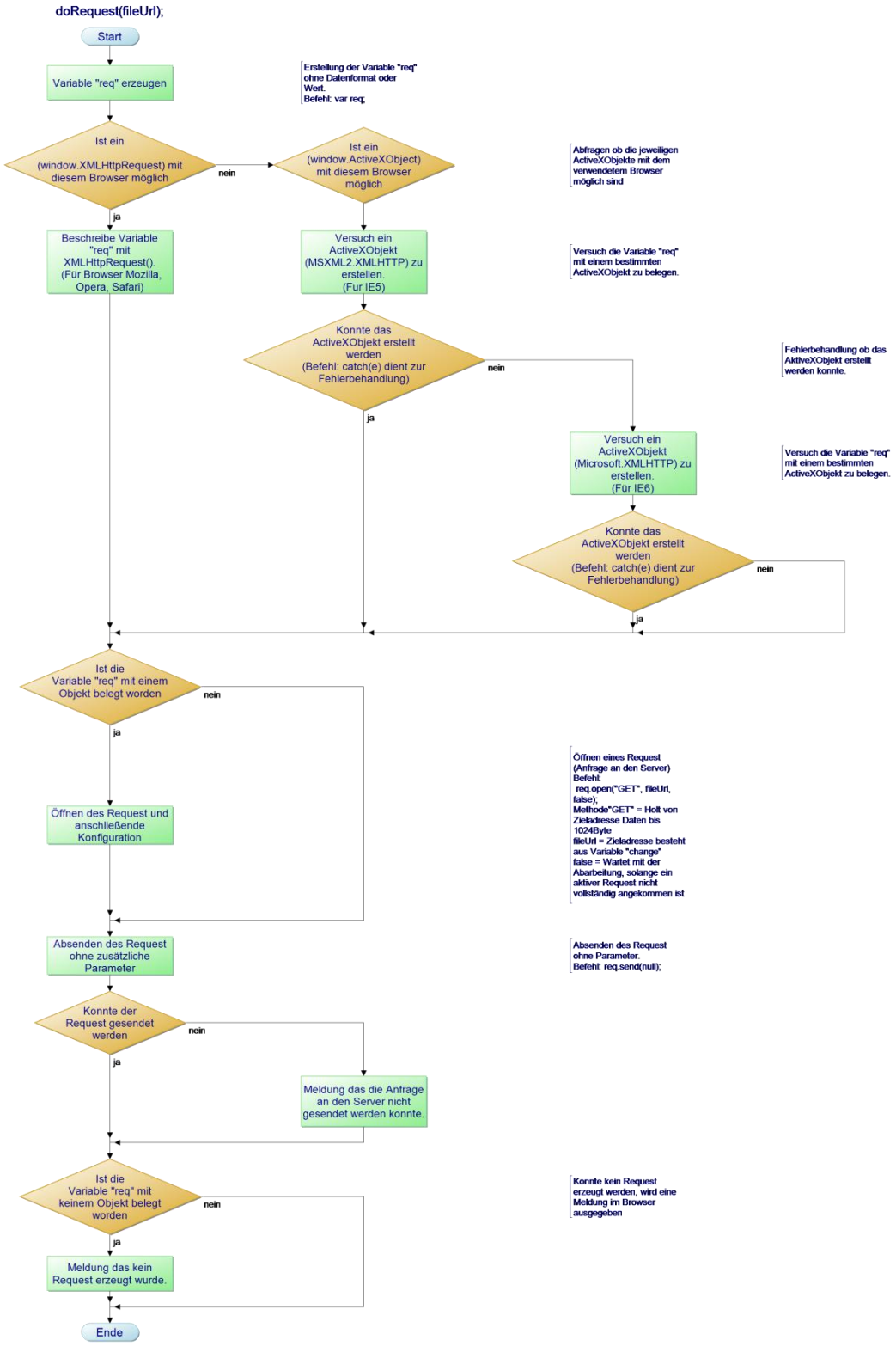


Abbildung 37 doRequest(fileUrl)

A.2 Quelltexte

Wie schon bei den Programmablaufplänen handelt es sich hier um eine zusammenhängende Gesamtübersicht, welche dem besseren Verständnis dienen soll.

A.2.1 myAVR MK2 Systemboard Hauptprogramm

```
//------------------------------------------------------------------------
//------------------------------------------------------------------------
//-------Einfügen der Bibliotheken----------------------------------------
//------------------------------------------------------------------------
//------------------------------------------------------------------------
#define  F_CPU 3686400
#define  MyEthernet 0xB0
#include <avr/io.h>
#include <avr/eeprom.h>
#include <inttypes.h>
#include "i2cmaster.h"
#include "rc5.h"

//------------------------------------------------------------------------
//------------------------------------------------------------------------
//-------Variablendeklaration---------------------------------------------
//------------------------------------------------------------------------
//------------------------------------------------------------------------

int empfangsfach=0;
uint16_t sendefach=0;
//int telegrammspeicher[32] EEMEM;
int zaehler=0;
uint16_t telegrammspeicher[32] EEMEM;       //Telegrammspeicher für 32 Einträge


char i2c_com_byte = 0;
/*Diese Variable ist für die i2c-Kommunikation mit dem MyAVR Ethernetboard
Der Inhalt dieser Variable ist von der zyklischen Abfrage eines Speicherbe-
reichs im MyAVR Ethernetboard abhängig. Der Ihalt dieser Variable wird als
folgendes Telegramm interpretiert:*/

/*************************************************************************
* i2c_com_byte Telegrammaufbau: |7|6|5|4|3|2|1|0|                        *
* Bit   7: Kontrollbit (Ist 1, wenn Telegramm vorliegt)                  *
* Bit   6: 1 -> Schreibe den Inhalt des Empfangsfachs in Telegrammspeicher*
*          0 -> Schreibe Inhalt des Telegrammspeichers in das Sendefach   *
* Bit 0-5: Feldadresse des Telegrammspeichers(0-32)                      *
*************************************************************************/
char i2c_com_byte_tmp;

//------------------------------------------------------------------------
//------------------------------------------------------------------------
//-------Unterprogramme---------------------------------------------------
//------------------------------------------------------------------------
//------------------------------------------------------------------------
void i2c_receive()
{

  i2c_start(MyEthernet+I2C_WRITE);        // Start, Schreiben und Slave-Adresse

  i2c_write(0x00);                        // SRAM-Adresse
  i2c_rep_start(MyEthernet+I2C_READ);     // widerholter Start, Lesen und
                                          // und Slave-Adresse
  i2c_com_byte_tmp = i2c_readNak();       // Lesevorgang aus dem SRAM

  i2c_stop();                             // Stoppkondition

  if (i2c_com_byte_tmp!=0)                // Speicherung nur bei Inhalt
```

```
  {
    i2c_com_byte=i2c_com_byte_tmp;
  }
}
//---------------------------------------------------------------------------
void i2c_clear_byte()
{

  i2c_start(MyEthernet+I2C_WRITE);        // Start, Schreiben und Slave-Adresse
  i2c_write(0x00);                        // SRAM-Adresse
  i2c_write(0x00);                        // zu schreibende Daten
  i2c_stop();                             // Stoppkondition

}
//---------------------------------------------------------------------------
void initialisierung()
{

  DDRB  |= (7<<2);

  DDRB  &= ~(1<<5);
  PORTB |=  (1<<5);

  DDRD  &= ~(1<<2);
  PORTD |=  (1<<2);



  i2c_master_init();
  PORTC |=  (3<<4);  //PullUp für TWI

  rc5_receive_init();
  rc5_send_init();

}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//-------Hauptprogramm-------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
int main (void)
{
  initialisierung();



  while(1)
  {
    /*********************************************************************
    *Auslesung des Bytes im MyAVR Ethernetboard                          *
    *********************************************************************/
    if (zaehler==30000)       // Inhalt wird ca. alle 412,5ms ausgeführt
    {
      i2c_receive();    // Speicherbereich im Ethernetboard wird ausgelesen

      if (i2c_com_byte!=0)
      {
        i2c_clear_byte(); // Speicherbereich im Ethernetboard wird mit null
                          // überschrieben (nur wenn sich ein Inhalt darin befand)
      }

      zaehler=0;
    }
    zaehler++;


    /*********************************************************************
    * Speicherung des Telegramms in Telegrammspeicher                    *
    *********************************************************************/
    if ((i2c_com_byte & 0xC0)==0xC0)
    {
      eeprom_write_word(&telegrammspeicher[i2c_com_byte&0x3F],empfangsfach);
```

```
      i2c_com_byte=0;
    }
    /*********************************************************************
    * Beschreibung des „sendefachs"s mit Telegramms                      *
    *********************************************************************/
    if ((i2c_com_byte & 0x80)==0x80)
    {
      sendefach=eeprom_read_word(&telegrammspeicher[i2c_com_byte&0x3F]);

      i2c_com_byte=0;
    }
    /*********************************************************************
    * Empfangsvorgang des rc5-Telegramms                                 *
    *********************************************************************/
    if (bit_is_clear(PINB,0))// Erkennung Telegrammbeginn
    {
     empfangsfach=rc5_receive();
    }
    /*********************************************************************
    * Sendevorgang des rc5-Telegramms                                    *
    *********************************************************************/
    if(sendefach>0) // Abfrage auf Inhalt in dem sendefach
    {
      rc5_send(sendefach);
      /*Da das RC5-Telegramm 14 Bit hat, das Sendefach jedoch 16, muss dieses
      nach dem Sendevorgang gelöscht werden, damit nicht ein erneuter sendevor-
      gang ausgelöst wird.*/

      sendefach=0;
    }
  }                          // Ende While (1)
}
```

A.2.2 myAVR MK2 Systemboard RC5 Bibliothek

Rc5.h

```
#ifndef _rc5_H
#define _rc5_H

extern void rc5_receive_init(void);
extern int rc5_receive(void);
extern void rc5_send_init(void);
extern void rc5_send(int data);

#endif
```

Rc5.c

```
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
//-------Festlegungen-------------------------------------------------------
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------

#define  F_CPU 3686400     // Taktfrequenz des myAVR-Boards

//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
//-------Einfügen der Bibliotheken------------------------------------------
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------

#include <avr/io.h>        // Grundbibliothek
#include <stdio.h>         // Bibliothek für Sprünge
```

```
#include "rc5.h"           // prototypen

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//-------Variablendeklaration------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
  int toggle_bit=0;
  /*Für die Realisierung einer Toggle_Funktin wird diese Variable benötigt,
  die nicht in der Funktion "void rc5_send(int data)" deklariert und ini-
  tialisiert werden, da sonst beim Aufruf der Funktion immer die 0 darin
  stehen und somit nicht getoggelt werden würde.*/
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//-------Funktionen----------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------

/*************************************************************************
* Initialisierung des Empfangsports:                                     *
*Diese Funktion initialisiert den Port B.0 als Eingang für den Empfang des*
*RC-5 Protokolls.                                                        *
*************************************************************************/
void rc5_receive_init()
{
  //Eingänge
  DDRB &= ~(1<<0);      //B.0 = Eingang
  PORTB|=  (1<<0);      //B.0 = PullUP
}


/*************************************************************************
*Empfangen eines Rc-5 Telegramms:                                        *
*Für die Verwendung dieser Funktion im Hauptprogramm wird                *
*die Verwendung folgender Syntax empfohlen:                              *
*                                                                       *
*if (bit_is_clear(PINB,0)) // Erkennung Telegrammbeginn                  *
*{                                                                       *
*  variable=rc5_receive();                                              *
*}                                                                       *
*************************************************************************/
int rc5_receive(void)
{
  /*In diese Variable wird das empfangene RC5 Telegramm vorläufig hineinge-
  schrieben, um anschließend zu testen, ob das Telegramm fehler-
  frei empfangen wurde.*/
  int empfangsfach=0;

  /*In diese Variable wird eine Kopie des PINB Registers geschrieben,um
  eine Flanke an dem Eingang des Empfängermuduls feststellen zu können.
  (sowohl eine negativ als auch eine positive Flanke)*/
  int kopie_pinb=0;

  /*Diese Variable wird ausschließlich dafür verwendet, Zuweisungen auszu-
  führen um Wartezeiten zu realisieren.*/
  int dummy=0;

  /*Dies ist eine Zählvariable, die für die Realisierung einer Wartezeit
  und einer Pulsweitenmodulation verwendet wird.*/
  int wartezeit=0;

  empfangsfach=1;    // Eintrag des 1. Startbits in
  anfang:
  for (wartezeit = 0;wartezeit<=213;wartezeit++)
                              // ca. 3/4 Periodendauer (1335µs)
  {
    dummy=1;
  }
  kopie_pinb=PINB & 1;       // Kopie für anschl. Fl.-Erkennung
  for (wartezeit= 0;wartezeit<=96;wartezeit++)
                                          // ca. 1/2 Periodendauer (889µs)
  {
```

```
    if ((kopie_pinb^(PINB & 1))==1)
    {
      if (kopie_pinb==0)
      {
        empfangsfach=empfangsfach*2;
      }

      if (kopie_pinb==1)
      {
        empfangsfach=empfangsfach*2;
        empfangsfach|=1;
      }

      goto anfang;
    }
  }

  if((empfangsfach&0x2000)==0x2000)
  {
    return (empfangsfach);
  }
  else
  {
    return 0x0001;
  }
}
/*****************************************************************************
* Initialisierung des Empfangsports:                                         *
*Diese Funktion initialisiert den Port B.0 als Eingang für den Empfang des*
*RC-5 Protokolls.                                                            *
*****************************************************************************/
void rc5_send_init()
{
  DDRB   |=  (1<<1);     //B.1= Ausgang für Sendediode
}
/*****************************************************************************
*Senden eines Rc5 Telegramms                                                 *
*Diese Funktion gibt das zu sendende Telegramm an PortB.1 aus                *
*                                                                            *
*ACHTUNG!!!! FUNKTIONIERT NUR MIT DEM ATMEGA8!!!                             *
*Da alle Zeiten in der Senderoutine auf des Taktfrequenz des ATmega8 ba-     *
*sieren und keine fertigen Wartezeiten verwendet werden, sind Trägerfre-     *
*quenz und Periodendauer auch nur mit dem ATmega8 (evtl. anderer Prozessor*
*mit gleicher Taktzeit) entsprechend dem RC5-Protokoll.                      *
*                                                                            *
*Für die Verwendung dieser Funktion im Hauptprogramm wird                    *
*die Verwendung folgender Syntax empfohlen:                                  *
*                                                                            *
*if(data>0) // Abfrage auf Inhalt in dem data                                *
*{                                                                           *
*  rc5_dend(variable);                                                       *
*                                                                            *
*  //Da das RC5-Telegramm 14 Bit hat, das data jedoch 16, muss dieses       *
*  //nach dem Sendevorgang gelöscht werden, damit nicht ein erneuter        *
*  //sendevorgang ausgelöst wird.                                           *
*  variable=0;                                                               *
*}                                                                           *
*****************************************************************************/
void rc5_send(int data)
{

  int gesendete_bits=0;
  int pwm_takt=0;
  int dummy=0;

  if ((toggle_bit&0x0001)==0x0001)
  /*Vergleich auf logisch 1 von dem toggle_bit*/
  {
    data|=0x0800;
    /*Ist der Wert des toggle_bit 's logisch 1, wird das entsprechnede
    Bit im Telegramm auf logisch 1 gesetzt.*/
    toggle_bit=0x0000;
```

```
}
else
{
  data&=0xF7FF;
  /*Ist der Wert des toggle_bit 's logisch 0, wird das entsprechnede
  Bit im Telegramm auf logisch 0 gesetzt.*/
  toggle_bit=0x0001;
}
while (gesendete_bits<14)
{
/* Solange nicht alle 14 Bits des Telegramms gesendet sind, wird
diese Schleife ausgeführt*/
  for(pwm_takt=0;pwm_takt<=133;pwm_takt++)
  {
  /*Über diese Schleife wird die Impus- bzw. Pausendauer der Bits im
  Telegramm realisiert. Der pwm_takt wird nachfolgend verwendet, um
  nachfolgende Pulsweitenmodulation zu realisieren*/
    if (pwm_takt<=63)
    {
    /*Solange der pwm_takt kleiner oder gleich 62 ist, wird die erste
    Hälfte der Pulsdauer realisiert*/
      if ((data&0x2000)==0x2000)
      /*Abfrage, ob das momentan zu sendete Bit den Wert Logisch 1
      besitzt*/
      {
        PORTB &=~(1<<1);
        /*Im Falle einer logischen 1, wird dem Ausgang für die Sende-
        diode in der ersten Hälfte der Bitdauer eine logische 0 zuge-
        wiesen*/
      }
      else
      {
        PORTB ^= (1<<1);
        /*Im Falle einer logischen 0, wird dem Ausgang für die Sende-
        diode in der ersten Hälfte der Bitdauer eine logische 1 zuge-
        wiesen*/
        dummy=0;
        /*Diese Zuweisung hat ausschließlich denn Sinn, Zeit in An-
        spruch zu nehmen, um die Abweichung der Trägerfrequenz zu
        minimieren*/
      }
    }
    else
    {
    /*Sobald der pwm_takt größer als 63 ist, wird die zweite Hälfte
    der Pulsdauer realisiert. Hirbei entsteht eine positive bzw. ne-
    gative Flange in der Hälfte, welche auf Grund der Kodierung nach
    dem Manchester-Verfahren benötigt wird. Eine positive Falnage
    entspricht einer logischen 1 und eine negative Flanke einer lo-
    gischen 0*/
      if ((data&0x2000)==0x2000)
      /*Abfrage, ob das momentan zu sendete Bit den Wert Logisch 1
      besitzt*/
      {
        PORTB ^= (1<<1);
        /*Im Falle einer logischen 1, wird dem Ausgang für die Sende-
        diode in der zweiten Hälfte der Bitdauer eine logische 1
        zugewiesen*/
        dummy=0;
        /*Diese Zuweisung hat ausschließlich denn Sinn, Zeit in An-
        spruch zu nehmen, um die Abweichung der Trägerfrequenz zu
        minimieren*/
      }
      else
      {
        PORTB &=~(1<<1);
        /*Im Falle einer logischen 0, wird dem Ausgang für die Sende-
        diode in der zweiten Hälfte der Bitdauer eine logische 0
        zugewiesen*/
      }
    }
  }
```

```
    gesendete_bits++;
    /*Da nun ein Bit gesendet wurde, wird die Anzahl der gesendeten
    Bits um 1 addiert*/
    data=(data<<1);
    /*Der Inhalt von data wird um ein Bit nach links geschoben,wo-
    durch das Bit, welches als nächstes gesendet werden soll, an die
    entsprechende Stelle in data geschoben wird.*/
  }
  data=0;
  /*Da das RC5-Telegramm 14 Bit hat, das data jedoch 16, muss dieses
  nach dem Sendevorgang gelöscht werden, damit nicht ein erneuter sendevor-
  gang ausgelöst wird.*/
  gesendete_bits=0;
  /*Die Variable in der die gesendeten Bits gezählt werden, wird wieder
  auf 0 zurückgesetzt*/
  //toggle_bit^=0x0001;
  /*Für den Fall, dass anschließend ein Telegramm mit dem selben Inhalt
  gesendet werden soll, wird das toggle_bit invertiert, damit der emp-
  fänger das wiederholte Senden erkennen kann*/
}
```

A.2.3 myEthernet Systemboard home.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Steuerungs-Homepage</title>  <!-- Titel der Homepage -->
    <meta content="Evrsoft First Page" name="GENERATOR">
    <link href="style.css" type="text/css" rel="stylesheet">  <!-- .css Datei -->
  </head>
  <body>
    <div id="div_content">  <!-- Content-Box -->
      <div id="div_banner">  <!-- Banner-Box -->
        <img src="Bilder/banner3.jpg" alt="banner">
      </div>
      <div id="div_navi">  <!-- Navigations-Box -->
        <ul>  <!-- Aufzaehlungsliste -->
          <li><a class="auswahl" href="home.htm">Home</a></li>
          <li> <a class="navi" href="tv.htm">TV</a></li>
        </ul>
      <br><br><br><br><br><br><br><br><br><br><br>
      <h1 class="news" >News</h1>  <!-- Überschrift 1.Ordnung -->
        <p  class="news">Unsere Projekte werden bald online gehen.</p>
      </div>
      <div id="div_inhalt">  <!-- Inhalt-Box -->
        <img src="Bilder/Ueberschrift_home.gif" class="ueberschrift">
        <p class="inhalt"> Willkommen in Ihrem Automatisiertem Zuhause! <br>
        <br>
        Steuern Sie gleich hier ihr gesamtes Haus über das Internet.<br>
        Noch ist unser Projekt in der Probe-Phase, aber bald werden wir <br>
        unsere Seite optimieren und Sie werden die Möglichkeit haben <br>
        auch Dinge, wie Ihre Heizung und Ihre Hifi Geräte über dieses <br>
        Webangebot steuern zu können.<br>
        Sehen Sie sich doch einfach mal um.</p><br>
      </div>
      <div id="div_footer">  <!-- Fußzeilen-Box -->
        <a class="impressum" href="impress.htm">Impressum</a>
      </div>
    </div>
  </body>
</html>
```

A.2.4 myEthernet Systemboard tv.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<script language="JavaScript" type="text/javascript">
  var modus=0;
  var kontrollbit=128;
  var senden=64;
  sendemodus = new Image();  //Erstelle neues Image "sendemodus"
  sendemodus.src = "bilder/sendemodus.jpg";  //Zuweisung der Bilddatei
  programmiermodus = new Image();  //Erstelle neues Image "programmiermodus"
  programmiermodus.src = "bilder/programmiermodus.jpg";  //Zuweisung der Bilddatei

  //Moduswechsel zwischen Programmier- und Sendemodus
  function moduswechsel()
  {
    modus = modus+1;  //Wird die Funktion aufgerufen wird "modus" um 1 inkrementiert
    if (modus==1)  //Vergleich ob "modus" = 1
    {
      document.images.modus.src = programmiermodus.src;    //Anzeige des Bildes "programmiermodus"
    }
    if (modus==2)  //Vergleich "modus"  = 2
    {
      document.images.modus.src = sendemodus.src;  //Anzeige des Bildes "sendemodus"
      modus=0;  //Variable "modus" auf 0 zurücksetzen
    }
  }

  //Schreibe in SRAM des myEthernet
  function schreibe_sram(taste) // Funktion für das Schreiben in den Shared Ram
  {
    if(modus==1)  //Ist Programmiermodus aktiv
    {
      var change="?myChangeCmd=%B0o1000%7E"+(taste+kontrollbit)+"%B0";  //Variable "change" wird mit
                                                                        //dem Kommando und dem Wert beschrieben
      doRequest (change);  //Unterprogramm "doRequest" für Ajax, übergabe von "change"
    }
    else  // Ist Sendemodus aktiv
    {
      var change="?myChangeCmd=%B0o1000%7E"+(taste+kontrollbit+senden)+"%B0";  //Variable "change"
                                              //wird mit dem Kommando und dem Wert beschrieben
      doRequest (change);  //Unterprogramm "doRequest" für Ajax, übergabe von "change"
    }
  }

  // Ajax Request
  function doRequest(fileUrl)
  { //fileUrl = myChangeCmd....
    var req;   //Variable für den Request erzeugen
    if (window.XMLHttpRequest)  //Kontrolliert ob ein XMLHttpRequest erlaubt ist
    {
      req = new XMLHttpRequest();  //erzeuge neuen Request für Mozilla,Opera,Safari
    } else if (window.ActiveXObject)  //Kontrolliert ob ein ActiveXObject erlaubt ist
    {
      try  //Versuche Einstellung für IE5
      {
        req = new ActiveXObject("MSXML2.XMLHTTP");  //erzeuge ActiveXObject für IE5
      }
```

```
      catch(e)  //Fehlerbehandlung. Wenn Browser nicht mit dem ActiveXObject("MSXML2.XMLHTTP")
      {          //umgehen kann
        try  //Versuche Einstellung für IE6
        {
          req = new ActiveXObject("Microsoft.XMLHTTP"); //erzeuge ActiveXObject für IE6
        }
        catch(e)  //Fehlerbehandlung. Wenn Browser nicht mit dem ActiveXObject("Microsoft.XMLHTTP")
        {          //umgehen kann
          ;
        }
      }
    }
    if (req) //Wenn Request erzeugt wurde
    {
      req.open("GET", fileUrl, true);  //Öffne Request: Methode=GET, fileUrl=Zieladresse, False=warte
    }                                  //bis aktiver Request vollständig
    Try
    {
      req.send(null);    //Absenden des Request ohne Parameter
    }
    catch(e)
    {
      alert("Es konnte keine Anforderung an den Server gesendet werden!");  //Fehlermeldung;
    }
    if (!req) //Konnte kein Request erzeugt werden
    {
      alert("Kann keine XMLHTTP-Instanz erzeugen!");  //Fehlermeldung
    }
  }
</script>

  <head>
    <title>Steuerungs-Homepage</title>
    <link media="screen" href="style.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <div id="div_content_tv">  <!-- Content-Box -->
      <div id="div_banner">  <!-- Banner-Box -->
        <img src="Bilder/banner2.jpg" alt="banner" />
      </div>
      <div id="div_navi">  <!-- Navigations-Box -->
        <ul>  <!-- Aufzählungsliste -->
          <li><a class="navi" href="home.htm">Home</a></li>
          <li> <a class="auswahl" href="tv.htm">TV</a></li>
        </ul>
        <br><br><br><br><br><br><br><br><br><br><br>
        <h1 class="news" >News</h1>  <!-- Überschrift 1.Ordnung -->
        <p  class="news">Unsere Projekte werden bald online gehen.</p>
      </div>
      <div id="div_inhalt">  <!-- Inhalt-Box -->
        <img src="Bilder/Ueberschrift_TV.png" class="ueberschrift">
        <img src="bilder/sendemodus.jpg" alt="Sendemodus" name="modus" class="modusgrafik">
        <form  class="button">     <!-- Erzeuge Windowsbutton -->
        <input onclick="javascript:moduswechsel();" type="button" value="Moduswechsel">
        </form>
        <img src="bilder/fernbedienung_hp.jpg"alt="Fernbedienung" usemap="#fernbedienung"
        class="fernbedienung">
        <!-- Mit dem Tag <map> wird eine Verweis-sensitive Grafik erzeugt. -->
        <!-- In der Eingebundenen Grafik können Kreise, Rechtecke oder Polygone bestimmt werden -->
        <!-- die einen Link enthalten. "shape" gibt die Geometrische Form des Verweises an -->
```

```
<!-- "coords" sind die Koordinaten an dem der Verweis in der Grafik aufhalten soll -->
<!-- Diesem Verweis wird mit "href" ein Standard-Link zugewiesen. -->
<!-- Mit dem "shape:poly" kann eine beliebige Geometrische Form erzeugt werden -->
<map name="fernbedienung">
  <area shape="circle" coords="83,30,15"
    href="javascript:schreibe_sram(10);" title="Ein/Aus" alt="Ein/Aus">
  <area shape="rect" coords="19,365,60,384"
    href="javascript:schreibe_sram(1);" title="Kanal 1" alt="Kanal 1">
  <area shape="rect" coords="62,365,100,384"
    href="javascript:schreibe_sram(2);" title="Kanal 2" alt="Kanal 2">
  <area shape="rect" coords="102,365,145,384"
    href="javascript:schreibe_sram(3);" title="Kanal 3" alt="Kanal 3">
  <area shape="rect" coords="19,392,60,411"
    href="javascript:schreibe_sram(4);" title="Kanal 4" alt="Kanal 4">
  <area shape="rect" coords="62,392,100,411"
    href="javascript:schreibe_sram(5);" title="Kanal 5" alt="Kanal 5">
  <area shape="rect" coords="102,392,145,411"
    href="javascript:schreibe_sram(6);" title="Kanal 6" alt="Kanal 6">
  <area shape="rect" coords="19,419,60,437"
    href="javascript:schreibe_sram(7);" title="Kanal 7" alt="Kanal 7">
  <area shape="rect" coords="62,419,100,437"
    href="javascript:schreibe_sram(8);" title="Kanal 8" alt="Kanal 8">
  <area shape="rect" coords="102,419,145,437"
    href="javascript:schreibe_sram(9);" title="Kanal 9" alt="Kanal 9">
  <area shape="rect" coords="62,446,100,464"
    href="javascript:schreibe_sram(0);" title="Kanal 0" alt="Kanal 0">
  <area shape="poly" coords="25,276,31,271,45,271,54,276,54,305,25,305"
    href="javascript:schreibe_sram(11);" title="Lautstärke +" alt="Laustärke +">
  <area shape="poly" coords="25,316,54,316,54,342,45,348,31,348,25,342"
    href="javascript:schreibe_sram(12);" title="Lautstärke -" alt="´Lautstärke -">
  <area shape="poly" coords="112,276,117,271,132,271,140,276,140,305,112,305"
    href="javascript:schreibe_sram(13);" title="Programm +" alt="Programm +">
  <area shape="poly" coords="112,316,140,316,140,342,132,348,117,348,112,342"
    href="javascript:schreibe_sram(14);" title="Programm -" alt="Programm -">
</map>
<br>
<p class="inhalt">Programmiermodus:<br>
<br>
Schritt 1:<br>
Halten Sie die Fernbedienung Ihres Fernsehers an den Empfänger des Mikrocontrollers.<br>
<br>
Schritt 2:<br>
Drücken Sie nun die Taste auf Ihrer Fernbedinung die Sie Einprogram-<br>
mieren möchten.<br>
<br>
Schritt 3:<br>
Ist der Einlesevorgang erfolgreich gewesen, können Sie nun eine gewünschte
Taste auf der Webfernbedienung drücken auf der das eingelesene Kommando
gespeichert werden soll.<br>
Auf dieser Webfernbedienung können folgende Tasten zum Speichern genutzt werden:<br>
<ul>    <!-- Aufzählungsliste -->
  <li><p class="liste">- Ziffern 0 - 9</p></li>
  <li><p class="liste">- Ein/Aus</p></li>
  <li><p class="liste">- Programm +/-</p></li>
  <li><p class="liste">- Lautstärke +/-</p></li>
</ul>
<p class="inhalt">Schritt 4:<br>
Nun kann mit Hilfe des "Moduswechsel" Buttons in den Sendemodus gewechselt
werden um Ihren Fernseher steuern zu können.</p>
</p>
```

```
      </div>
      <div id="div_footer">    <!-- Fußzeilen-Box -->
        <a class="impressum" href="impress.htm">Impressum</a>
      </div>
    </div>
  </body>
</html>
```

A.2.5 myEthernet Systemboard impress.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Steuerungs-Homepage</title>
    <meta content="Evrsoft First Page" name="GENERATOR" />
    <link media="screen" href="style.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
    <div id="div_content">  <!-- Content-Box -->
      <div id="div_banner">  <!-- Banner-Box -->
        <img src="Bilder/banner3.jpg" alt="banner" />
      </div>
      <div id="div_navi">  <!-- Navigations-Box -->
        <ul>    <!-- Aufzaehlungsliste -->
         <li><a class="navi" href="home.htm">Home</a></li>
          <li> <a class="navi" href="tv.htm">TV</a></li>
        </ul>
        <br><br><br><br><br><br><br><br><br><br>
        <h1 class="news" >News</h1>    <!-- Ueberschrift 1. Ordnung -->
        <p  class="news">Unsere Projekte werden bald online gehen.</p>
      </div>
      <div id="div_inhalt">  <!-- Inhalt-Box -->
        <img src="Bilder/Ueberschrift_impressum.gif" class="ueberschrift">
        <p class="inhalt">
          Manuel Gutekunst<br>
          Luxemburger Straße 8<br>
          54294 Trier<br>
          manuel.gutekunst@gmx.de
          <br><br><br><br><br>
          Benjamin Gräbedünkel<br>
          Rodestraße 10<br>
          54290 Trier<br>
          benni_graebeduenkel@web.de
          <br><br><br><br><br>
          Sämtliche Inhalte dieses Internetauftritts unterliegen<br>
          dem Copyright.
          </p><br>
      </div>
      <div id="div_footer">  <!-- Fußzeilen-Box -->
        <a class="impressum" href="impress.htm">Impressum</a>
      </div>
    </div>
  </body>
</html>
```

A.2.6 myEthernet Systemboard style.css

```
body    /* Hintergrundfarbe für den Tag <body> */
{
  background-color: #447aa3;
}

/*Textklassen-----------------------------------------------------------------------------------------*/

p    /* Formatierung für den Tag <p> */
{
  font-size: 12px;
  font-family:Verdana, Arial, Helvetica, sans-serif;
}

h1    /* Ueberschrift 1.Ordnung */
{
  font-family: "Times New Roman", Times, serif;
  font-size:14px;
  padding-top: 50px;
  color: #0f436b;
}

.liste
{
  font-family:Arial, Helvetica, sans-serif;
}

.inhalt    /* Schriftformatierung für <p class="inhalt">*/
{
  font-family:Arial, Helvetica, sans-serif;
  padding-right: 50px;
  padding-top: 30px;
}

.news    /* Schriftformatierung für <p class="news"> */
{
  padding-left: 20px;
}

.ueberschrift    /* Schriftformatierung für <p class="ueberschrift"> */
{
  padding-top: 30px;
}

/*Grafikklassen---------------------------------------------------------------------------------------*/

.fernbedienung    /* Grafikformatierung für <img class="fernbedienung"> */
{
  float: right;
  padding-right: 20px;
  width: 160 px;
  height:550 px;
  border: none;
}

.button    /* Grafikformatierung für <form class="button"> */
{
  float: right;
```

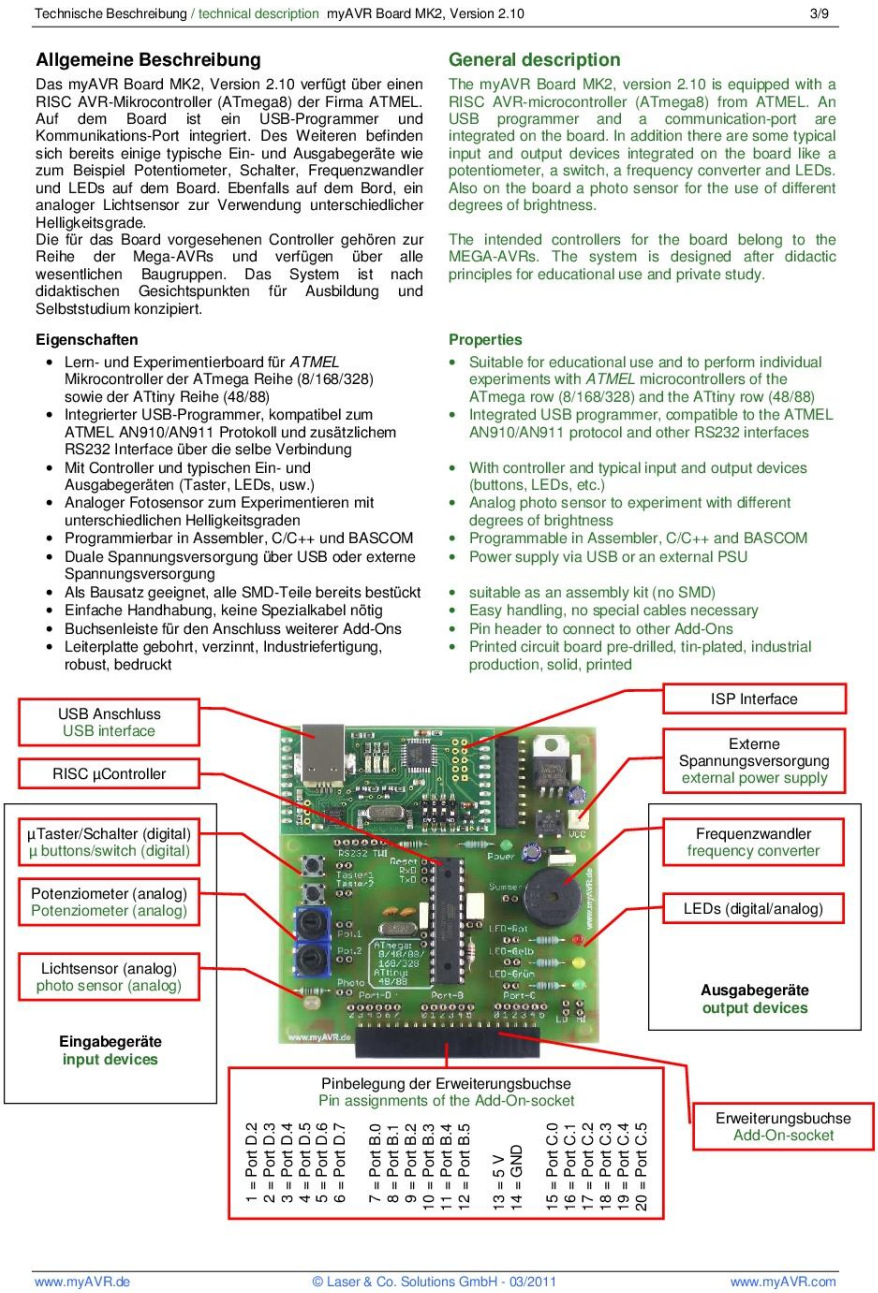
```
  padding-right: 45px;
  padding-top: 20px;
  margin: 0px;
}

.modusgrafik    /* Grafikformatierung für <img class="modusgrafik"> */
{
  float: left;
  width: 280px;
  height: 45px;
}

/*LInks-------------------------------------------------------------------------------------------*/

a    /* Textformatierung für den Tag <a> */
{
  text-decoration:none;
  font-family:Verdana, Arial, Helvetica, sans-serif;
  font-size:12px;
  padding-bottom:20px;
  padding-left: 50px;
  color: #666666;
  text-align: center;
  display: block;
}

a:hover    /* Textformatierung wenn die Maus über einen Link bewegt wird */
{
  color: #2f02ca;
  font-weight:bold;
}

a.navi    /* Textformatierung für Navigationstext <a class="navi"> */
{
  text-decoration:none;
  font-family:Verdana, Arial, Helvetica, sans-serif;
  font-size:12px;
  padding-bottom:20px;
  padding-left: 50px;
  color: #666666;
  text-align: center;
  display: block;
}

a.auswahl    /* Textformatierung für <a class="auswahl"> */
{
  color: #245d93;
  background-image:url(Bilder/Link_auswahl.gif);
  display: block;
  background-repeat:no-repeat;
}

a.impressum    /* Textformatierung für <a class="impressum"> */
{
  color:#333333;
  font-size: 9px;
  padding-left: 10px;
  float:left;
}
/*Boxen-------------------------------------------------------------------------------------------*/
```

```
#div_content    /*Hauptbox*/
{
  margin-left: 220px;    /*Aussenabstand links*/
  margin-top: 5%;        /*Aussenabstand oben*/
  width: 800px;          /*Breite der Box*/
  height: 900px;         /*Hoehe der Box*/
  background-image: url(Bilder/hintergrund.jpg);    /*Hintergrundbild*/
}

#div_content_tv    /*Hauptbox für TV*/
{
  margin-left: 220px;    /*Aussenabstand links*/
  margin-top: 5%;        /*Aussenabstand oben*/
  width: 800px;          /*Breite der Box*/
  height: 900px;         /*Hoehe der Box*/
  background-image: url(Bilder/Hintergrund_tv.jpg);    /*Hintergrundbild*/
}

#div_banner     /*Bannerbox*/
{
  width: 800px;
  height: 200px;
}

#div_navi    /*Navigationsbox*/
{
  padding-top: 25px;
  float:left;
  width: 125px;
  height: 650px;
}

#div_inhalt    /*Inhaltbox*/
{
  float:right;
  width: 600px;
  height: 675px;
  padding-left:50px;
}

#div_footer    /*Fussbox*/
{
  float: left;
  width: 800px;
  height: 25px;
  text-align:left;
}
/*Listen-------------------------------------------------------------------------------------------*/
ul    /* Listenformatierung für den Tag <ul>  */
{
  list-style: none;
  padding-top: 0px;
  padding-left: 0px;
}

li.navi    /* Textformatierung für Listeneinträge in der Navigation <li> */
{
  float:left;
  padding-bottom: 10px;
}
```

Anhang B

B.1 Datenblätter

B.1.1 myAVR Board MK2, bestückt



Technische Beschreibung / technical description myAVR Board MK2, Version 2.10 3/9

Allgemeine Beschreibung

Das myAVR Board MK2, Version 2.10 verfügt über einen RISC AVR-Mikrocontroller (ATmega8) der Firma ATMEL. Auf dem Board ist ein USB-Programmer und Kommunikations-Port integriert. Des Weiteren befinden sich bereits einige typische Ein- und Ausgabegeräte wie zum Beispiel Potentiometer, Schalter, Frequenzwandler und LEDs auf dem Board. Ebenfalls auf dem Bord, ein analoger Lichtsensor zur Verwendung unterschiedlicher Helligkeitsgrade.
Die für das Board vorgesehenen Controller gehören zur Reihe der Mega-AVRs und verfügen über alle wesentlichen Baugruppen. Das System ist nach didaktischen Gesichtspunkten für Ausbildung und Selbststudium konzipiert.

Eigenschaften

- Lern- und Experimentierboard für *ATMEL* Mikrocontroller der ATmega Reihe (8/168/328) sowie der ATtiny Reihe (48/88)
- Integrierter USB-Programmer, kompatibel zum ATMEL AN910/AN911 Protokoll und zusätzlichem RS232 Interface über die selbe Verbindung
- Mit Controller und typischen Ein- und Ausgabegeräten (Taster, LEDs, usw.)
- Analoger Fotosensor zum Experimentieren mit unterschiedlichen Helligkeitsgraden
- Programmierbar in Assembler, C/C++ und BASCOM
- Duale Spannungsversorgung über USB oder externe Spannungsversorgung
- Als Bausatz geeignet, alle SMD-Teile bereits bestückt
- Einfache Handhabung, keine Spezialkabel nötig
- Buchsenleiste für den Anschluss weiterer Add-Ons
- Leiterplatte gebohrt, verzinnt, Industriefertigung, robust, bedruckt

General description

The myAVR Board MK2, version 2.10 is equipped with a RISC AVR-microcontroller (ATmega8) from ATMEL. An USB programmer and a communication-port are integrated on the board. In addition there are some typical input and output devices integrated on the board like a potentiometer, a switch, a frequency converter and LEDs. Also on the board a photo sensor for the use of different degrees of brightness.

The intended controllers for the board belong to the MEGA-AVRs. The system is designed after didactic principles for educational use and private study.

Properties

- Suitable for educational use and to perform individual experiments with *ATMEL* microcontrollers of the ATmega row (8/168/328) and the ATtiny row (48/88)
- Integrated USB programmer, compatible to the ATMEL AN910/AN911 protocol and other RS232 interfaces
- With controller and typical input and output devices (buttons, LEDs, etc.)
- Analog photo sensor to experiment with different degrees of brightness
- Programmable in Assembler, C/C++ and BASCOM
- Power supply via USB or an external PSU
- suitable as an assembly kit (no SMD)
- Easy handling, no special cables necessary
- Pin header to connect to other Add-Ons
- Printed circuit board pre-drilled, tin-plated, industrial production, solid, printed

www.myAVR.de © Laser & Co. Solutions GmbH - 03/2011 www.myAVR.com

Abbildung 38 Datenblatt myAVR MK2, bestückt Seite 1 **(myavr.de 2011)**

USB-Programmer und Interface

Der USB-Programmer ist fertig bestückt (SMD-Bauweise) und wird als Tochterplatine auf das myAVR Board MK2 USB gesteckt. Er ist einzeln erhältlich.
Für mehr Informationen, lesen Sie bitte die technische Beschreibung zum USB-Programmer mySmartUSB MK2.

USB programmer and interface

The USB programmer is fully equipped (SMD technology) and is used with the myAVR Board MK2 USB as a daughterboard. It is also available separatly.
Please read the technical description of the USB programmer mySmartUSB MK2 for more information.

Technische Daten		Technical Data	
Betriebsdaten		**Operating Data**	
Versorgungsspannung	empfohlen 9 V stabilisierte Gleichspannung	Supply Voltage	9 V stabilised DC voltage recommended
Betriebsstrom	10-50 mA typisch ohne weitere Verbraucher	Operating Current	10-50 mA, typical without other loads
Betriebsspannung	3,3 – 5,3	Operating Voltage	3.3 – 5.3 V
Betriebstemperatur	0 °C bis +30 °C	Operating Temperature	0 °C to +30 °C
Lagertemperatur	-20 °C bis +70 °C	Storage Temperature	-20 °C up to +70 °C
Strom	Beachte USB-Spezifikation in der Regel 100mA	Current	Take into account, USB specification of as a rule 100 mA
Schnittstellendaten		**Interface Data**	
Programmierung und Kommunikation über USB 2: USB-Buchse für Anschluss an PC mit Standard-USB-Kabel A-B		programming and communication via USB 2: USB-pin for connection with PC via Standard-USB-cable A-B	

Mechanische Daten		Mechanical Data	
myAVR Board		**myAVR Board**	
Abmaße (L x B x H)	ca. 90 mm x 90 mm x 18 mm	Dimensions (L x W x H)	ca. 90 mm x 90 mm x 18 mm
Gewicht	ca. 50 g	Weight	ca. 50 g
Rastermaß	2,54 mm	Grid dimensions	2.54 mm
Tochterplatine		**Daughterboard**	
Abmaße (L x B x H)	ca. 60 mm x 30 mm x 15 mm	Dimensions (L x W x H)	ca. 60 mm x 30 mm x 15 mm
Gewicht	ca. 30 g	Weight	ca. 30 g
Rastermaß	2,54 mm, für Komponenten in Printmontage	Grid dimensions	2.54 mm, for components in print assembly
USB-Controller	CP2102, SiliconLabs	USB controller	CP2102, SiliconLabs

Abbildung 39 Datenblatt myAVR MK2, bestückt Seite 2 **(myavr.de 2011)**

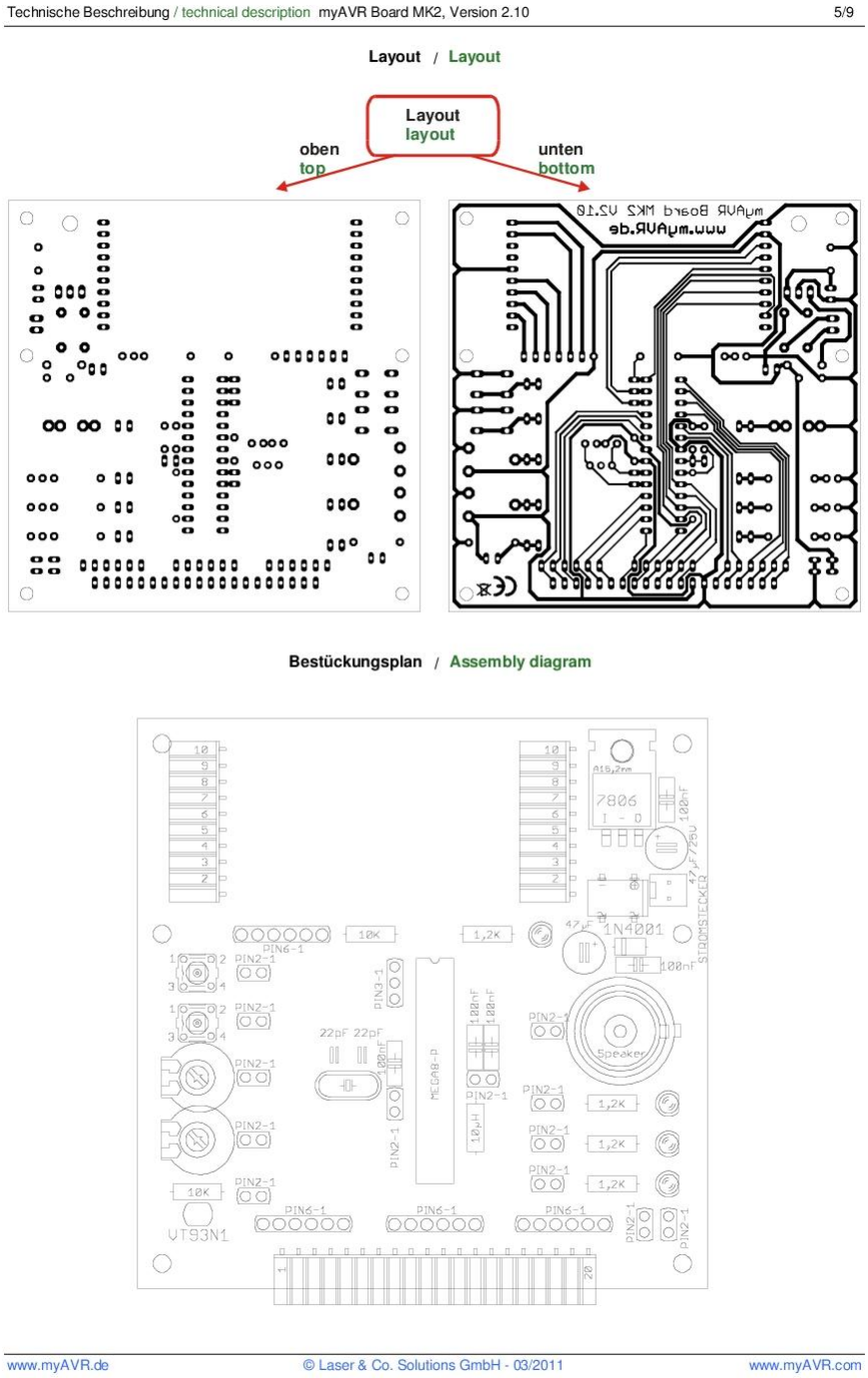


Abbildung 40 Datenblatt myAVR MK2, bestückt Seite 3 **(myavr.de 2011)**

Schaltplan / Circuit diagram

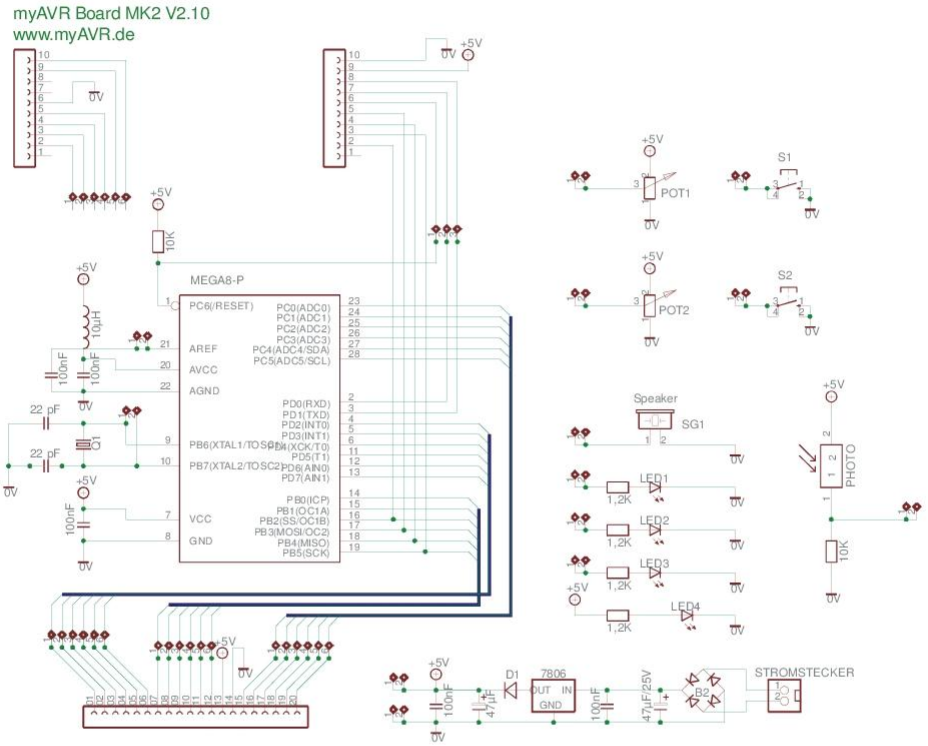


Abbildung 41 Datenblatt myAVR MK2, bestückt Seite 4 **(myavr.de 2011)**

B.1.2 myEthernet

1 Produkteigenschaften

- 10 MegaBit Ethernet mit ENC28J60 von Microchip
- ATmega644P 20MHz mit vorinstalliertem Webserver
  - IP-Adresse konfigurierbar (192.168.20.96 = standard)
  - 64 K FLASH
  - 4 K Byte SRAM
  - 2 K Byte EEPROM
- MicroSD-Kartenhalter
- Ethernet Buchse mit Übertrager
- ISP-Anschluss 10polig
- TWI / UART / SPI-Schnittstelle
- Erweiterungsbuchse nach myAVR-Standard
- Qualitätsleiterplatte FR4, Industriefertigung, robust, bedruckt

1 Product features

- 10 megabit Ethernet with ENC28J60 from Microchip
- ATmega644P 20 MHz with preinstalled web server
  - IP address configurable (192.168.20.96 = standard)
  - 64 K FLASH
  - 4 KByte SRAM
  - 2 Kbyte EEPROM
- microSD card holder
- ethernet connector with transformer
- ISP connection 10 pin
- TWI / UART / SPI interface
- add-on socket with myAVR standard
- printed circuit board FR4, industrial production, solid

1.1 Technische Daten	
Betriebsdaten	
Betriebsspannung	5 V, max. 5,5 V
Betriebsstrom	< 350 mA
Betriebstemperatur	0 – 30 °C
Lagertemperatur	-20 °C – 70 °C

1.1 Technical data	
Operating Data	
Operating voltage	5 V, max. 5,5 V
Operating current	< 350 mA
Operating temperature	0 – 30 °C
Storage temperature	-20 °C – 70 °C

1.2 Mechanische Daten	
Abmaße (L x B x H)	90 x 30 x 19 mm
Masse	~20 g
Rastermaß	2,54 mm

1.2 Mechanical data	
Dimensions (L x W x H)	90 x 30 x 19 mm
Weight	~20 g
Grid dimensions	2.54 mm

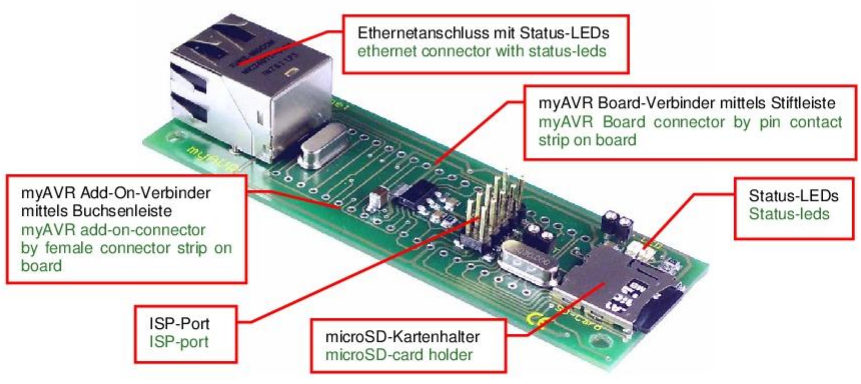
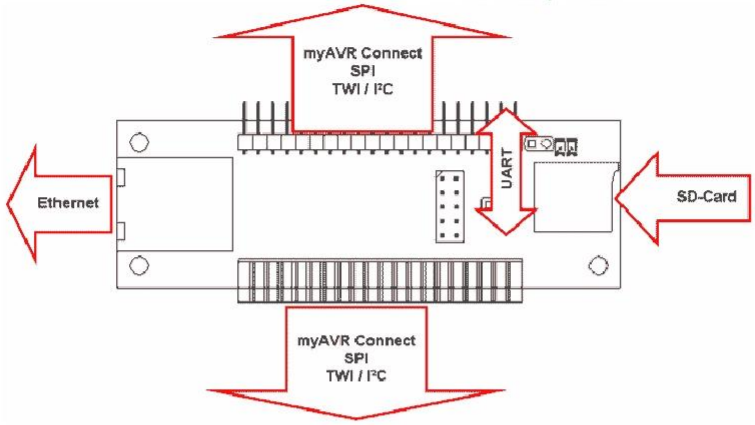


Abbildung 42 Datenblatt myEthernet Seite 1 **(myavr.de 2011)**

2 Technische Details

2.1 Funktionen

2.1.1 Schnittstellen / Protokolle



Pinübergreifende Funktionalität:
Der Mikrocontroller ATmega644P bietet neben der bitweisen Ein / Ausgabe auch eine byteweise Ein / Ausgabe über 8 Pins, diese werden zusammengefasst auch als PortA / B bezeichnet.

TWI / I²C:
Ist ein serieller Datenbus, der über 2 Leitungen kommunziert. Er erlaubt die Adressierung von bis zu 128 Geräten.

SPI - Serial Peripheral Interface:
Ist ein Bussystem für eine synchrone serielle Datenübertragung. Dabei werden 3 Leitungen für MOSI (Master out Slave in), MISO (Master in Slave out), SCK (Serial Clock) und eine für SS (Slave Select) verwendet.

UART - Universal Asynchronous Receiver Transmitter:
Serielle Kommunikation ohne Taktsignal. Empfänger und Sender synchronisieren sich durch Start- und Stopp-Bit.

ISP - In-System-Programmierung:
Programmieren Sie direkt im Einsatzsystem ohne den zu programmierenden Mikrocontroller herauszunehmen.

MicroSD-Karte:
Um Zugriff auf eine Vielzahl an Dateien sowie einen robusten Datenspeicher zu besitzen, verfügt das myEthernet-Board über eine Schnittstelle für MicroSD-Karten. Diese wird über den Mikrocontroller ATmega644P gesteuert.

2 Technical details

2.1 Functions

2.1.1 Interfaces / protocols

Pin functionality across:
The microcontroller ATmega644P offers between the bit by bit input / output also a byte by byte input / output over 8 pins. These 8 pins were also summarised to portA / B.

TWI / I²C:
That's a seral data bus which communicates via two lines. He alows the addressing up to 128 devices.

SPI – Serial Peripheral Interface:
Thats a bus system for a synchronous serial data communication. Thereby three lines will be used for MOSI (Master out Slave in), MISO (Master in Slave out), SCK (Serial Clock) and one for SS (Slave Select).

UART – Universal Asynchronous Receiver Transmitter:
A serial communication without clock signal. Receiver and transmitter synchronise them through a start and stop bit.

ISP in system programming:
Programming directly in the using system without taking out the microcontroller which is programmed.

microSD-card:
To have access to a multiplicity of files and a robust data memory the myEthernet has an interface for microSD-cards. This interface will be controlled through the microcontroller Atmega644P.

Abbildung 43 Datenblatt myEthernet Seite 2 **(myavr.de 2011)**

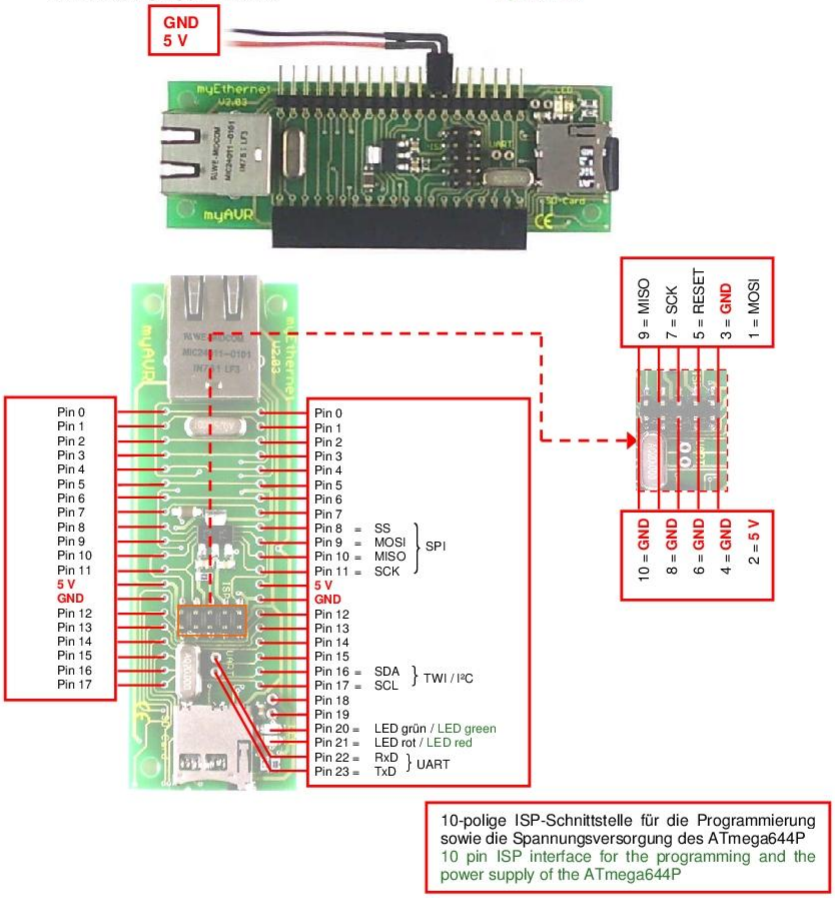
myEthernet Technische Beschreibung / technical description 7/48

myEthernet:

Die Kommunikation über das Netzwerk wird mit dem Mikrocontroller ENC28J60 realisiert. Der Datentransfer des embedded Webservers wird durch das Protokoll HTTP umgesetzt. Als unterliegende Protokolle arbeiten dabei das Transportprotokoll TCP, das Vermittlungsprotokoll IP, ICMP und in der untersten Ebene die Bitübertragung per Ethernet.

myEthernet:

The communication via the network will be realized with the microcontroller ENC28J60. The data transfer of the embedded web server will be implemented through the protocol HTTP. As underlying protocols work the transport protocol TCP, the exchange protocol IP and in the first-line level the bit transfer via myEthernet.



10-polige ISP-Schnittstelle für die Programmierung sowie die Spannungsversorgung des ATmega644P
10 pin ISP interface for the programming and the power supply of the ATmega644P

Pin	0	1	2	3	4	5	6	7	8	9	10	11	12
Port am ATmega 644P	A4	A5	A6	A7	B0	B1	B2	B3	B4	B5	B6	B7	A0

Pin	13	14	15	16	17	18	19	20	21	22	23
Port am ATmega644P	A1	A2	A3	C1	C0	C2	C3	C4	C5	D0	D1

www.myAVR.de © Laser & Co. Solutions GmbH – 06/2010 www.myAVR.com

Abbildung 44 Datenblatt myEthernet Seite 3 **(myavr.de 2011)**

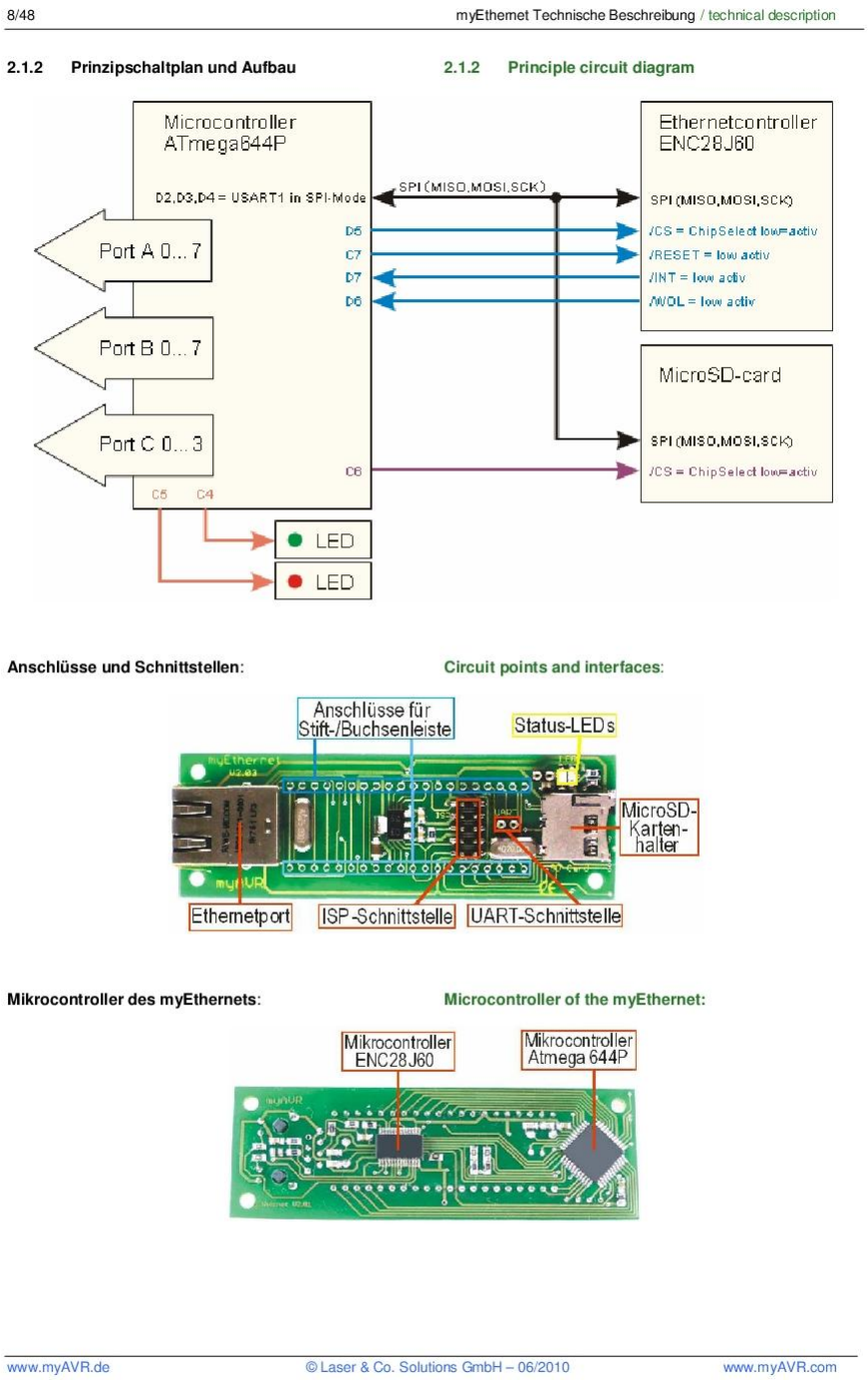


Abbildung 45 Datenblatt myEthernet Seite 4 **(myavr.de 2011)**

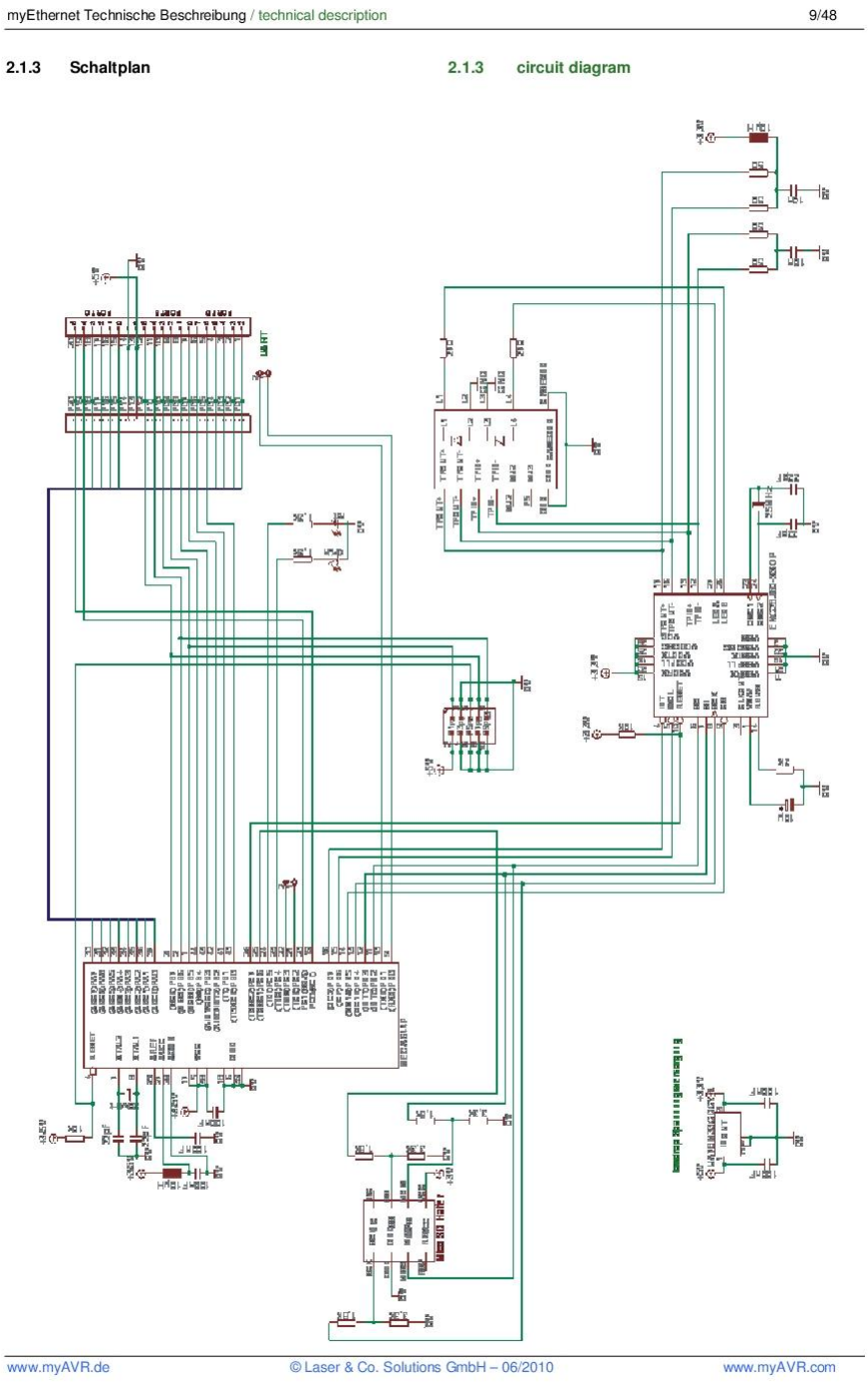


Abbildung 46 Datenblatt myEthernet Seite 5 **(myavr.de 2011)**

B.1.3 IR-Empfängermodul TSOP 4836

VISHAY

TSOP48..

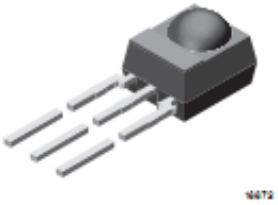
Vishay Semiconductors

IR Receiver Modules for Remote Control Systems

Description

The TSOP48.. - series are miniaturized receivers for infrared remote control systems. PIN diode and preamplifier are assembled on lead frame, the epoxy package is designed as IR filter.

The demodulated output signal can directly be decoded by a microprocessor. TSOP48.. is the standard IR remote control receiver series, supporting all major transmission codes.



16473

Features

- Photo detector and preamplifier in one package
- Internal filter for PCM frequency
- Improved shielding against electrical field disturbance
- TTL and CMOS compatibility
- Output active low
- Low power consumption

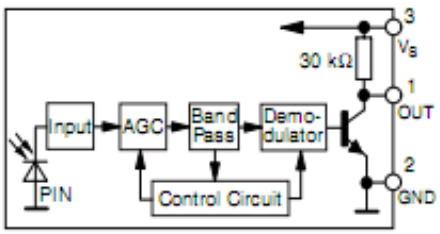
Special Features

- Improved immunity against ambient light
- Suitable burst length ≥ 10 cycles/burst

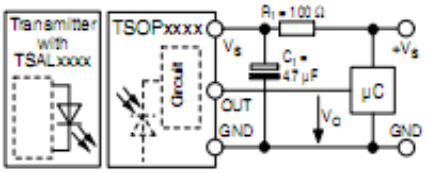
Parts Table

Part	Carrier Frequency
TSOP4830	30 kHz
TSOP4833	33 kHz
TSOP4836	36 kHz
TSOP4837	36.7 kHz
TSOP4838	38 kHz
TSOP4840	40 kHz
TSOP4856	56 kHz

Block Diagram



Application Circuit



R_1 + C_1 recommended to suppress power supply disturbances.

The output voltage should not be hold continuously at a voltage below V_O = 3.3 V by the external circuit.

Absolute Maximum Ratings

T_{amb} = 25 °C, unless otherwise specified

Parameter	Test condition	Symbol	Value	Unit
Supply Voltage	(Pin 3)	V_S	-0.3 to + 6.0	V

Abbildung 47 Datenblatt TSOP Empfängermodul Seite1 **(alldatasheet.com 2011)**

TSOP48..

Vishay Semiconductors

VISHAY

Parameter	Test condition	Symbol	Value	Unit
Supply Current	(Pin 3)	I_S	5	mA
Output Voltage	(Pin 1)	V_O	-0.3 to +6.0	V
Output Current	(Pin 1)	I_O	5	mA
Junction Temperature		T_j	100	°C
Storage Temperature Range		T_{stg}	-25 to +85	°C
Operating Temperature Range		T_{amb}	-25 to +85	°C
Power Consumption	(T_{amb} ≤ 85 °C)	P_{tot}	50	mW
Soldering Temperature	t ≤ 10 s, 1 mm from case	T_{sd}	260	°C

Electrical and Optical Characteristics

T_{amb} = 25 °C, unless otherwise specified

Parameter	Test condition	Symbol	Min	Typ.	Max	Unit
Supply Current (Pin 3)	V_S = 5 V, E_v = 0	I_{SD}	0.8	1.2	1.5	mA
	V_S = 5 V, E_v = 40 klx, sunlight	I_{SH}		1.5		mA
Supply Voltage		V_S	4.5		5.5	V
Transmission Distance	E_v = 0, test signal see fig.1, IR diode TSAL6200, I_F = 250 mA	d		35		m
Output Voltage Low (Pin 1)	I_{OSL} = 0.5 mA, E_e = 0.7 mW/m², test signal see fig. 1	V_{OSL}			250	mV
Irradiance (56 kHz)	Pulse width tolerance: t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o, test signal see fig.1	$E_{e\,min}$		0.3	0.5	mW/m²
Irradiance (30-40 kHz)	Pulse width tolerance: t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o, test signal see fig.1	$E_{e\,min}$		0.2	0.4	mW/m²
Irradiance	t_{pi} - 5/f_o < t_{po} < t_{pi} + 6/f_o, test signal see fig. 1	$E_{e\,max}$	30			W/m²
Directivity	Angle of half transmission distance	$\varphi_{1/2}$		±45		deg

Typical Characteristics (T_{amb} = 25°C unless otherwise specified)

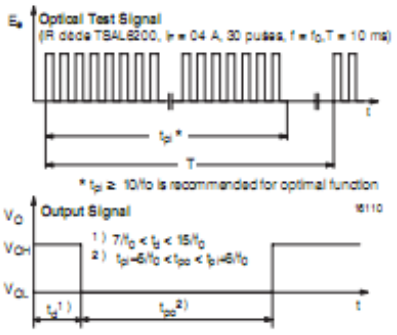


Figure 1. Output Function

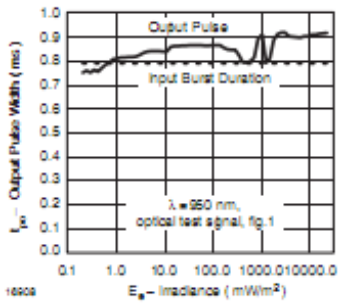


Figure 2. Pulse Length and Sensitivity in Dark Ambient

Abbildung 48 Datenblatt Empfängermodul Seite2 **(alldatasheet.com 2011)**

B.1.4 IR-Sendediode TSUS520

VISHAY

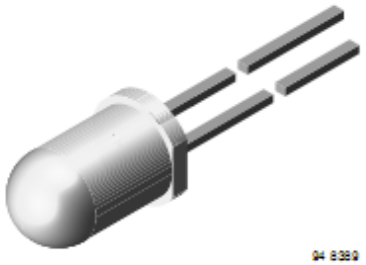
TSUS520.

Vishay Semiconductors

Infrared Emitting Diode, 950 nm, GaAs

Description

TSUS520. series are infrared emitting diodes in standard GaAs on GaAs technology, molded in a clear, blue-grey tinted plastic package. The devices are spectrally matched to silicon photodiodes and phototransistors.



94 8389

Features

- Low cost emitter
- Low forward voltage
- High radiant power and radiant intensity
- Suitable for DC and high pulse current operation
- Standard T-1¾ (∅5 mm) package
- Angle of half intensity $\varphi = \pm 15°$
- Peak wavelength $\lambda_p = 950$ nm
- High reliability
- Good spectral matching to Si photodetectors
- Lead (Pb)-free component
- Component in accordance to RoHS 2002/95/EC and WEEE 2002/96/EC

Applications

- Infrared remote control and free air transmission systems with low forward voltage and low cost requirements in combination with PIN photodiodes or phototransistors.

Absolute Maximum Ratings

T_{amb} = 25 °C, unless otherwise specified

Parameter	Test condition	Symbol	Value	Unit
Reverse voltage		V_R	5	V
Forward current		I_F	150	mA
Peak forward current	$t_p/T = 0.5$, $t_p = 100$ µs	I_{FM}	300	mA
Surge forward current	$t_p = 100$ µs	I_{FSM}	2.5	A
Power dissipation		P_V	210	mW
Junction temperature		T_j	100	°C
Operating temperature range		T_{amb}	- 55 to + 100	°C
Storage temperature range		T_{stg}	- 55 to + 100	°C
Soldering temperature	t ≤ 5 sec, 2 mm from case	T_{sd}	260	°C
Thermal resistance junction/ ambient		R_{thJA}	375	K/W

Abbildung 49 Datenblatt IR-Sendediode TSUS520 **(alldatasheet.com 2011)**

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur- und Quellverzeichnis

Stefan Buchgeher 2011 BUCHGEHER, Stefan: *RC5-Protokollaufbau*. RC5 (Dekodierung mit PIC-Mikrocontroller), 2004 – Bild Nr. 1.1, URL: http://www.stefan-buchgeher.info/elektronik/rc5/rc5_doku.pdf (2011-03-26)

alldatasheet.com 2011 IR-SENDEDIODE TSUS520 DES HERSTELLERS VISHAY: URL: http://html.alldatasheet.co.kr/html-pdf/183624/VISHAY/TSUS520_07/219/1/TSUS520_07.html (2011-04-21)

IR-EMPFÄNGERMODUL TSOP4836 DES HERSTELLERS VISHAY: URL: http://www.alldatasheet.com/datasheet-pdf/pdf/252285/VISHAY/TSOP4836.html (2011-04-21)

wikipedia.de 2011 WIKIPEDIA DIE FREIE ENZYKLOPÄDIE: *I²C*, URL: http://de.wikipedia.org/wiki/I%C2%B2C (2011-04-29)

projekte.myavr.de 2011 AUFBAU DES SHARED RAM DES MYETHERNET SYSTEMBOARDS. URL: http://projekte.myavr.de/index.php?sp=pages/myethernet_shared_ram_download (2011-04-27)

myavr.de 2011 SYSTEMBOARD MYAVR MK2, BESTÜCKT DES HERSTELLERS LASER & CO SOLUTIONS GMBH: URL: http://www.myavr.info/download/produkte/myavr_board_mk2/techb_myavr-board-mk2_de_en.pdf (2011-04-27)

SYSTEMBOARD MYETHERNET DES HERSTELLERS LASER & CO SOLUTIONS GMBH: URL: http://www.myavr.info/download/produkte/myethernet/techb_myethernet_de_en.pdf (2011-04-27)