```
/*-----------------------------------------------------------------
  Beschreibung:
  Das Programm betreibt einen Webserver, mit dessen Hilfe man
  and den Arduino angeschlossene LEDs ueber einen Browser steuern kann.
  Die LEDs werden an die Digitalpins ledPin1 und ledPin2 angeschlossen
  Bei Bedarf hier <--***  vor dem Setup aendern
  Als Basis diente das unten erwaehnte Programm. Es wurde von
  Norbert Kraemer geaendert und verbessert

  Program:      eth_websrv_LED

  Description:  Arduino web server that serves up a web page
                allowing the user to control an LED

  Hardware:     - Arduino Uno and official Arduino Ethernet
                  shield. Should work with other Arduinos and
                  compatible Ethernet shields.
                - LED and resistor in series connected between
                  Arduino pin 2 and GND

  Software:     Developed using Arduino 1.0.3 software
                Should be compatible with Arduino 1.0 +

  References:   - WebServer example by David A. Mellis and
                  modified by Tom Igoe
                - Ethernet library documentation:
                  http://arduino.cc/en/Reference/Ethernet

  Date:         11 January 2013

  Author:       W.A. Smith, http://startingelectronics.org

-----------------------------------------------------------------*/

#include <SPI.h>
#include <Ethernet.h>
// #include <WiFi.h>

// MAC Adresse des Ethernet Shields
```

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// Aendern Sie das letzte Byte (0xED) in 0xF0, 0xF1, 0xF2, ...
IPAddress ip(192, 168, 178, 25); // IP Adresse, dem eigenen Netzwerk anpassen  <--***---- Hier eingreifen!
EthernetServer server(80);  // create a server at port 80
// WiFiServer server(80);
// Bei WiFi diese Zeile auskommentieren und die darueberliegende einkommentieren

const int anzAnalogPin = 3;       // Anzahl der auszugebenden Analogpins       <--***---- Hier eingreifen!
const int anzLED = 4;             // Anzahl der LEDs oder Relais, maximal 4    <--***---- Hier eingreifen!
int ledPin[] = { 2, 3, 5, 6 };    // LEDs oder Relais an den Pins 2, 3, 5, 6   <--***---- Hier eingreifen!
                                  // Beachte: Die Pins 4, 10, 11, 12 und 13 sind tabu!!!
const int anzEingabePin = 3;      // Anzahl der digitalen Eingabepins festlegen       <--***---- Hier eingreifen!
int eigabePin[] = { 7, 8, 9 } ;   // Hier werden die digitalen Eingabepins festgelegt.<--***---- Hier eingreifen!


int einPegel = LOW;           // Bei LEDs ist einPegel = HIGH, bei Relais ist einPegel = LOW, ggf. anpassen
int ausPegel = HIGH;
int LEDnr;

boolean LED_status[anzLED];   // Status von LED1 oder Relais1, Voreinstellung AUS (= ausPegel)
int i, i0,i1,i2,i3,j;         // Hilfsvariablen
String HTTP_req = "";         // Zeichenkette zur Speicherung der HTTP-Anfrage, Voreinstellung Leerstring
String Information;
boolean currentLineIsBlank;   // Hilfsvariable, um das Ende der HTTP-Anfrage zu erkennen
char c;                       // Hilfsvariable, um ein gelesenes Zeichen zwischenzuspeichern

String hilfsSc, hilfsSuc;
String hilfsSli ="<input type=\"checkbox\" name=\"L";
String hilfsSreuc ="\" value=\"1\" onclick=\"submit();\" >LED oder Relais ";
String hilfsSrec ="\" value=\"1\" onclick=\"submit();\" checked>LED oder Relais ";
String lhS;

// Diese Variablenvereinbarungen muessen vor dem Setup stehen!



void setup()
{
  // SD-Karte Slave Select SS ausschalten, wichtig beim Betrieb ohne SD-Karte
```

```
  pinMode(4, OUTPUT);
  digitalWrite(4, einPegel);

  Ethernet.begin(mac, ip);  // Erstmalige Inbetriebnahme des Ethernet Shields
  server.begin();           // Server einschalten und auf Anfrage warten (Lauschstellung)
  Serial.begin(9600);       // Seriellen Port fuer Ueberwachungszwecke oeffnen
  for (i=0; i<anzLED; i++) {
    pinMode(ledPin[i], OUTPUT);
    digitalWrite(ledPin[i], ausPegel);
    LED_status[i]=ausPegel;
  }
  for (i=0; i<anzEingabePin; i++) {
    pinMode(eigabePin[i], INPUT);
    digitalWrite(eigabePin[i], HIGH);
  }

}

void loop()
{
    EthernetClient client = server.available();  // Versuche Verbindung mit dem Client (=anfragenden Browser) aufzunehmen

    if (client) {                           // Wenn sich ein Client gemeldet hat
        currentLineIsBlank = true;          // Noch steht nichts in der (empfangenen) neuen Zeile
        while (client.connected()) {        // Solange der Client angeschlossen ist
          if (client.available()) {         // Wenn noch Zeichen vom Client empfangen werden koennen
            c = client.read();              // Lies ein einzelnes Zeichen
            HTTP_req += c;                  // Fuege dem AnfrageString das Zeichen hinzu
            //  ----------------------------------------------------------------
            // Was bis zu den naechsten --- steht, wird nur einmalig am Ende der HTTP-Anfrage ausgefuehrt
            // Solange der Server (=Arduino) noch Zeichen vom Client empfaengt, wird dieser Teil uebersprungen
            if (c == '\n' && currentLineIsBlank) {        // Ein Zeilenvorschub "\n", gefolgt von einer leeren Zeile ist das Ende
              // Das Ende der Anfrage ist nun erkannt, das Lauschen ist zu Ende, jetzt ist der Server (=Arduino) dran
              // Sende einen Standard http response header (Standard Antwortkopf)
              // Der Kopf ist kein Seiteninhalt, sondern sagt dem Browser nur, nach welchen Regeln er die Sache darzustellen hat
              client.println("HTTP/1.1 200 OK");          // Ich (Arduino) sende nun (an den Browser) nach dem HTTP/1.1 Standard
              client.println("Content-Type: text/html");  // Der Inhalt dessen, was der Arduino sendet, ist HTML-Text
              client.println("Connection: close");        // Nach Beendigung der Antwort wird der Server (=Arduino) die Verbindur
              client.println();
```

```
// Mit dieser Leerzeile ist der Antwortkopf abgeschlossen. Jetzt folgt der eigentliche Seiteninhalt

client.println("<!DOCTYPE html>");
client.println("<html>");
client.println("<head><title>Arduino LED Control</title></head>");
client.println("<body>");
client.println("<h2>Digitale Ein-/Ausgabe, analoge Eingabe</h2>");
client.println("<p>Die Pins 4 (SD-Card), 10, 11, 12, 13 sind f&uuml;r das Ethernetshield reserviert! <br>Also dies
client.println("<p>Klicken Sie, um die LED ein- und auszuschalten.</p>");
client.print("<p>LEDs oder Relais an den Pins ");
for (i=0; i<anzLED; i++) {
  client.print(ledPin[i]);
  client.print(",  ");
}
client.println("</p>");
client.println("<form method=\"get\">");  // action=\"123456\" , evtl. post statt get

// Nun werden die Checkboxen korrekt dargestellt (mit oder ohne Haken)
for (i=0; i<anzLED; i++) {
  VerarbeiteCheckbox(client, i);
}
client.println("</form>");
client.println("<br><br>");
for (i=0; i<anzAnalogPin; i++) {
  abfrageAnalogPin(client, i);
}
client.println("<br><br>");
client.println("Zwischen den Digitalpins und GND kann ein Schalter gelegt werden.<br>");
client.println("Bei offenem Schalter wird 1, bei geschlossenem 0 ausgegeben.<br><br>");

for (i=0; i<anzEingabePin; i++) {
  abfrageDigitalPin(client, eigabePin[i]);
}
client.println("<br><br>");
client.println("Neue Abrage durch Neuladen der Seite im Browser");
client.println("</body>");
client.println("</html>");
delay(100);           // gib dem Browser ein bisschen Zeit
```

```
            Kontrollausgabe2(HTTP_req);

            HTTP_req = "";        // HTTP-Anfrage abgearbeitet, String wieder leeren
            break;                // Die while-Schleife gewaltsam wieder verlassen
          }                       // Ende "if (c == '\n' && currentLineIsBlank)"
          //  ------------------------------------------------------------

          // Jede vom Client empfangene Textzeile endet mit \r\n ("\r" = Wagenruecklauf (reurn), "\n" = neue Zeile)
          if (c == '\n') {
            // Das Zeichen fuer neue Zeile wurde empfangen, d.h. noch ist die neue Zeile leer
            currentLineIsBlank = true;
          }
          else if (c != '\r') {     // Das Zeilenende ist noch nicht erreicht,
            // also war c ein Textzeichen, und damit ist die Zeile nicht leer
            currentLineIsBlank = false;
          }

          }          // Ende von "if (client.available())"
        }            // Ende von "while (client.connected())"

        delay(5);      // Gib dem Web-Browser Zeit, die Daten zu empfangen
        client.stop(); // Schliesse die Verbindung zum anfragenden Browser
    }                  // Ende von "if (client)"
}                      // Ende des Loop



// Jetzt kommen die selbst definierten Funktionen

// Schalte die LEDs und sende den passenden HTML-Code fuer die Checkboxen der LEDs an den anfragenden Browser
void VerarbeiteCheckbox(EthernetClient cl, int LEDnr) {
  i=LEDnr;
    Information="";
    i0=HTTP_req.indexOf("?");
    i1=HTTP_req.indexOf("HTTP");
    for (j=i0+1;j<i1-1;j++) {
      Information += HTTP_req.charAt(j);
    }
```

```
//Kontrollausgabe1(Information, i0,i1);    // Kann evtl. auskommentiert werden

//Hilfsstrings zusammenbasteln
  hilfsSc=hilfsSli;
  hilfsSc+=i;
  hilfsSc+=hilfsSrec;
  hilfsSc+=i+1;

  hilfsSuc=hilfsSli;
  hilfsSuc+=i;
  hilfsSuc+=hilfsSreuc;
  hilfsSuc+=i+1;

  lhS="L";
  lhS+=i;
  lhS+="=1";

if (Information.indexOf("GET") > -1) {               // Wenn das Wort GET in der Information vorkommt
  // Sende den HTML-Code fuer die Checkbox der LED[i] mit/ohne Haekchen an den anfragenden Browser
  if (LED_status[i] == 1) {
    cl.println(hilfsSc);
  }
  else {
    cl.println(hilfsSuc);
  }
  cl.println("      ");  // Ein paar Leerzeichen einfuegen
} // end if GET

else {
  // Setze den Ausgang fuer LED[i] am entsprechenden Pin auf den richtigen Wert
  if (Information.indexOf(lhS) > -1) {  // LED[i] anzeigen und schalten
      // the checkbox was clicked, toggle the LED
        LED_status[i] = 1;
        digitalWrite(ledPin[i], einPegel);
        // checkbox is checked
        cl.println(hilfsSc);
    }
    else {
```

```
            LED_status[i] = 0;
            digitalWrite(ledPin[i], ausPegel);
            // checkbox is unchecked
            cl.println(hilfsSuc);
          }
        cl.println("      ");  // Ein paar Leerzeichen einfuegen
      } // end else if GET
} // ende verarbeite Checkbox

void abfrageAnalogPin(EthernetClient cl, int i) {
  String hs="Analogpin ";
  hs+=i;
  hs+=" : ";
  cl.println(hs);
  cl.println(analogRead(i));
  cl.println("      ");  // Ein paar Leerzeichen einfuegen
}

void abfrageDigitalPin(EthernetClient cl, int i) {
  String hs="Digitalpin ";
  hs+=i;
  hs+=" : ";
  cl.println(hs);
  cl.println(digitalRead(i));
  cl.println("      ");  // Ein paar Leerzeichen einfuegen
}

void Kontrollausgabe1(String Information, int i0, int i1) {
  Serial.print("Die Zeichenkette ? beginnt ab Stelle ");
  Serial.println(i0);
  Serial.print("Die Zeichenkette HTTP beginnt ab Stelle ");
  Serial.println(i1);
  Serial.println("------- Uns interessiert nur diese Information -------");
  Serial.print(Information);
  Serial.println();
}

void Kontrollausgabe2(String HTTP_req) {
  Serial.println("Beginn der HTTP-Anfrage:");
```

```
  Serial.print(HTTP_req);
  Serial.print("Laenge der Anfrage: ");
  Serial.println(HTTP_req.length());
  Serial.println("-----<><><><><><><><><><>----");
}
```