```
/*-----------------------------------------------------------------
  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
EthernetServer server(80);  // create a server at port 80
// WiFiServer server(80);
// Bei WiFi diese Zeile auskommentieren und die darueberliegende einkommentieren

int ledPin1 = 2;            // LED1 on pin 2   <--***
int ledPin2 = 5;            // LED1 on pin 3   <--***
                            // Beachte: Die Pins 4, 10, 11, 12 und 13 sind tabu!!!
boolean LED1_status = 0;    // Status von LED1, Voreinstellung AUS (= 0)
boolean LED2_status = 0;    // Status von LED2, Voreinstellung AUS (= 0)
int 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
// 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, HIGH);

  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
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);

}

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 Verbindu
      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("<h1>Digitale Ein-/Ausgabe</h1>");
      client.println("<p>Die Pins 4 (SD-Card), 10, 11, 12, 13 sind f&uuml;r das Ethernetshield reserviert!</p>");
      client.println("<p>Klicken Sie, um die LED ein- und auszuschalten.</p>");
      client.print("<p>LED1 an Pin ");
      client.print(ledPin1);
      client.print(" und LED2 an Pin ");
      client.print(ledPin2);
      client.println("</p>");
      client.println("<form method=\"get\">");  // action=\"123456\" , evtl. post statt get
      // Nun werden die Checkboxen korrekt dargestellt (mit oder ohne Haken)
      ProcessCheckbox(client);
      client.println("</form>");
      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 ProcessCheckbox(EthernetClient cl) {
    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);
    if (Information.indexOf("GET") > -1) {               // Wenn das Wort GET in der Information vorkommt
      // Sende den HTML-Code fuer die Checkbox der LED1 mit/ohne Haekchen an den anfragenden Browser
      if (LED1_status == 1) {
```

```
      cl.println("<input type=\"checkbox\" name=\"L1\" value=\"1\" onclick=\"submit();\" checked>LED1");
    }
    else {
      cl.println("<input type=\"checkbox\" name=\"L1\" value=\"1\" onclick=\"submit();\">LED1");
    }

    // Sende den HTML-Code fuer die Checkbox der LED2 mit/ohne Haekchen an den anfragenden Browser
    if (LED2_status == 1) {
      cl.println("<input type=\"checkbox\" name=\"L2\" value=\"1\" onclick=\"submit();\" checked>LED2");
    }
    else {
      cl.println("<input type=\"checkbox\" name=\"L2\" value=\"1\" onclick=\"submit();\">LED2");
    }
  } // end if GET

  else {
    // Setze den Ausgang fuer LED1 am entsprechenden Pin auf den richtigen Wert
    if (Information.indexOf("L1=1") > -1) {  // LED1 anzeigen und schalten
        // the checkbox was clicked, toggle the LED
          LED1_status = 1;
          digitalWrite(ledPin1, HIGH);
          // checkbox is checked
          cl.println("<input type=\"checkbox\" name=\"L1\" value=\"1\" onclick=\"submit();\" checked>LED1");
      }
      else {
          LED1_status = 0;
          digitalWrite(ledPin1, LOW);
          // checkbox is unchecked
          cl.println("<input type=\"checkbox\" name=\"L1\" value=\"1\" onclick=\"submit();\">LED1");
      }
    cl.println("   ");

    // Setze den Ausgang fuer LED2 am entsprechenden Pin auf den richtigen Wert
    if (Information.indexOf("L2=1") > -1) {  // LED2 anzeigen und schalten
        // the checkbox was clicked, toggle the LED
          LED2_status = 1;
          digitalWrite(ledPin2, HIGH);
          // checkbox is checked
          cl.println("<input type=\"checkbox\" name=\"L2\" value=\"1\" onclick=\"submit();\" checked>LED2");
```

```
        }
        else {
            LED2_status = 0;
            digitalWrite(ledPin2, LOW);
            // checkbox is unchecked
            cl.println("<input type=\"checkbox\" name=\"L2\" value=\"1\" onclick=\"submit();\">LED2");
        }
      cl.println("   ");
    } // end else if GET
}



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("-----<><><><><><><><><><><>----");
}
```