Rechnernetze

Programmieren für Ingenieure
Sommer 2015

Andreas Zeller, Universität des Saarlandes

Algorithmus

- eindeutige *Handlungsvorschrift* zur Lösung eines Problems
- besteht aus endlich vielen, wohldefinierten *Einzelschritten*
- werden typischerweise in *Computerprogrammen* implementiert

Algorithmen

Berechnen	Suchen	Sortieren
• Fibonacci • GgT • Collatz	• Linear • Binär	• Einfügen • Mischen

Sortieren vor Ort

a

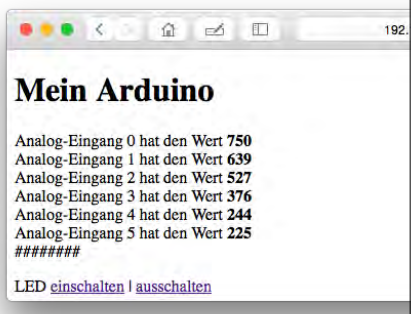
0	1	2	3	4	5	6	7	8	9	10
10	–10	7	2	2	–4	–7	–10	1	4	5

- Wir möchten *innerhalb des Feldes* sortieren
- Annhme: Feld a[0…i–1] ist bereits sortiert
- Wir betrachten das Element a[i]…
- …und fügen es in das sortierte Feld ein

Theremin =

Themen heute

- Rechnernetze
- Netzadressen
- HTTP
- HTML
- Webserver!



Murray Leinster

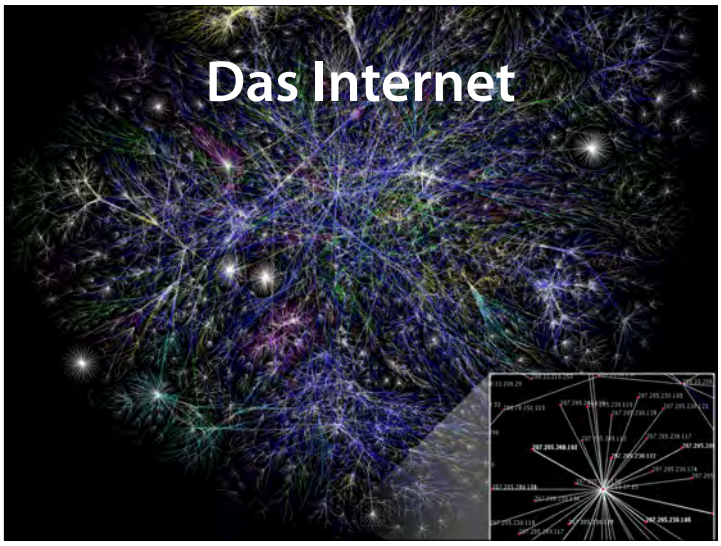
"A Logic Named Joe" (1946)

Murray Leinster, 1896–1975

Der Computer ... erledigt die Verbreitung von vierundneunzig Prozent aller Fernsehprogramme, vermittelt alle Informationen über Wetter, Luftverkehr, Sonderangebote … und dokumentiert jedes geschäftliche Gespräch, jeden Vertrag … Die Computer haben die Welt verändert. Die Computer sind die Zivilisation. Wenn wir die Computer abschalten, fallen wir in eine Art von Zivilisation zurück, von der wir vergessen haben, wie sie geht.

Ziel eines Rechnernetzes ist es, dass Rechner miteinander kommunizieren

---



Teile einer "Karte" des Internets, basierend auf Daten von opte.org am 15.01.2005. Jede Linie beschreibt zwei Knotenpunkte, welche zwei IP-Adressen repräsentieren. Die Länge der Linien beschreibt die Verzögerung zwischen den Knotenpunkten. Diese Karte beschreibt weniger als 30% der Klasse C Netzwerke, welche Anfang 2005 von dem Datensammelprogramm erreicht werden konnte. Die Linien sind farblich entsprechend der RFC 1918 Addressbereiche gekennzeichnet: Dunkelblau: net,

---

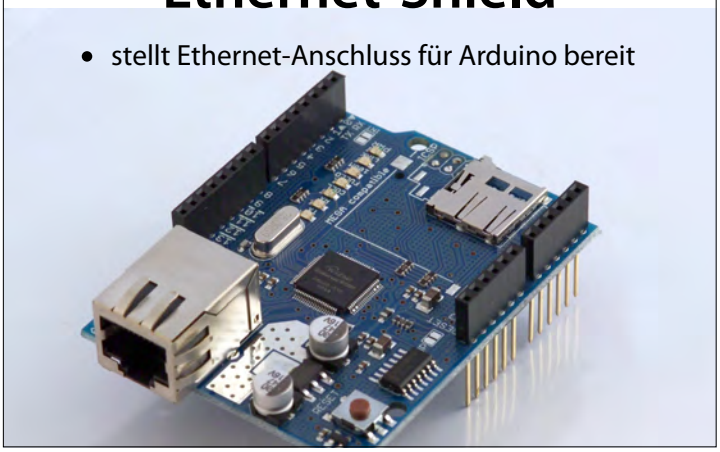
Ethernet

- Physikalische Datenverbindung zwischen zwei Geräten

Der Ethernet-Standard definiert Software und Hardware für kabelgebundene Datennetze für bis zu 100 Gbit/s. Ursprünglich für lokale Netze; Ethernet über Glasfaser hat eine Reichweite von 10 km und mehr.

Ethernet-Shield

- stellt Ethernet-Anschluss für Arduino bereit



MAC-Adresse

- identifiziert Geräte im Netz
- besteht aus sechs Zahlen von 0…255
- wird für jedes Gerät nur 1x vergeben
- Zahlen in *Hexadezimal-Format* (Basis 16)

Beispiel – `0A:F0:DE:AD:BE:EF`

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Die MAC-Adresse (Media-Access-Control-Adresse) ist die Hardware-Adresse jedes einzelnen Netzwerkadapters, die als eindeutiger Identifikator des Geräts in einem Rechnernetz dient.

MAC auf Arduino

- MAC-Adresse frei (aber eindeutig) wählbar
- Feld aus sechs Bytes (0…255), Typ "byte"
- Hexadezimalzahlen mit Präfix "0x"

```
byte mac[] = {
    0xDE, 0xAD, 0xAB, 0xBA, 0xFE, 0xED
};
```

entspricht – `DE:AD:AB:BA:FE:ED`

IP-Adresse

- IP = Internet Protocol
- Über die *IP-Adresse* wird ein Gerät in einem Netz eindeutig identifiziert
- Besteht aus 4 Bytes (IPv4)

Beispiel – `192.168.0.42`

Der Nachfolger von IPv4 ist IPv6, wobei die Version 6 hierbei keine überarbeitete Fassung des TCP-Protokolls mehr kennzeichnet. Seine wichtigste Modifizierung ist das Wenden-System. IPv4 verwendet 32-Bit-Adressen (ca. 4 Milliarden, oder 4,3·109 Adressen), während IPv6 128-Bit-Adressen verwendet (ca. 340 Sextillionen, oder 3,4·1038 Adressen).

Verbindung aufbauen

- Wir geben http://192.168.0.42/ im Browser ein
- Der Browser sucht nach einem Rechner mit dieser IP-Adresse
- Bereich 192.168.x.x für lokale Netze reserviert
- Andere Bereiche sind im Internet vergeben

Geben wir www.spiegel.de im Browser ein, übersetzt der Browser diese Adresse in eine numerische (IP)-Adresse im Internet.

IP-Adresse auf Arduino

- IP-Adresse muss in einem Netz eindeutig sein
- Wird mit "IPAddress ip(n_1, n_2, n_3, n_4)" angelegt

```
#include <SPI.h>
#include <Ethernet.h>

IPAddress ip(192, 168, 0, 42);
```

legt `192.168.0.42` als IP-Adresse an

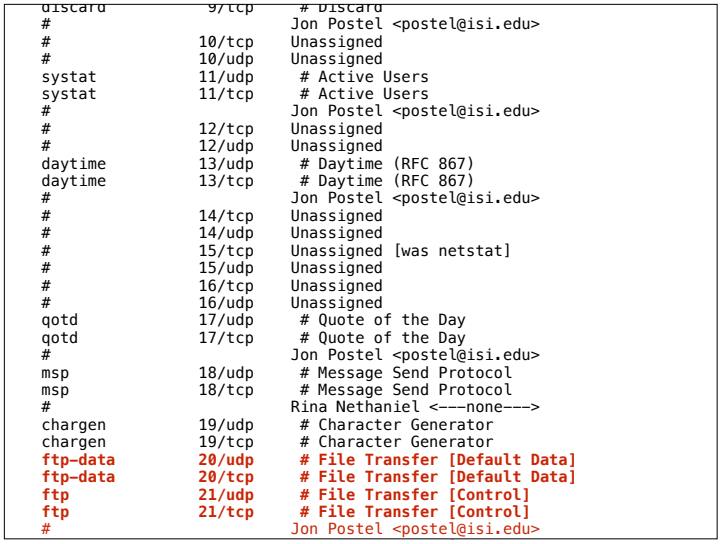
Ports

- Jeder Rechner stellt verschiedene *Ports* für IP-Verbindungen zur Verfügung
- Ports sind von 1…65535 nummeriert
- Jeder Dienst hat einen eigenen Port

Ports

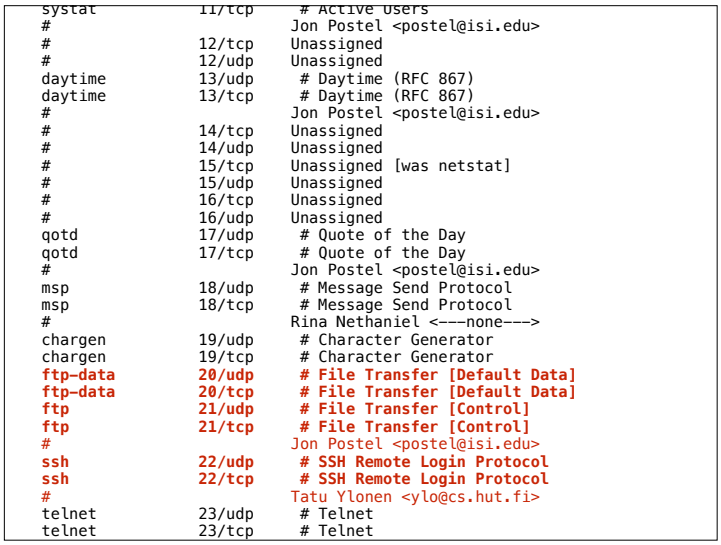
```
#   http://www.iana.org/assignments/port-numbers
#
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# $FreeBSD: src/etc/services,v 1.89 2002/12/17 23:59:10 eric Exp $
#   From: @(#)services   5.8 (Berkeley) 5/9/91
#
# WELL KNOWN PORT NUMBERS
#
rtmp              1/ddp    #Routing Table Maintenance Protocol
tcpmux            1/udp     # TCP Port Service Multiplexer
tcpmux            1/tcp     # TCP Port Service Multiplexer
#                           Mark Lottor <MKL@nisc.sri.com>
nbp               2/ddp    #Name Binding Protocol
compressnet       2/udp     # Management Utility
compressnet       2/tcp     # Management Utility
compressnet       3/udp     # Compression Process
compressnet       3/tcp     # Compression Process
#                           Bernie Volz <VOLZ@PROCESS.COM>
echo              4/ddp    #AppleTalk Echo Protocol
#                 4/tcp    Unassigned
#                 4/udp    Unassigned
rje               5/udp     # Remote Job Entry
rje               5/tcp     # Remote Job Entry
#                           Jon Postel <postel@isi.edu>
zip               6/ddp    #Zone Information Protocol
#                 6/tcp    Unassigned
#                 6/udp    Unassigned
echo              7/udp     # Echo
```

```
mit-ml-dev       85/udp     # MIT ML Device
mit-ml-dev       85/tcp     # MIT ML Device

... 12000 lines more ...



#                           David Reed <--none--->
#               47809-47999 Unassigned
nimcontroller   48000/udp    # Nimbus Controller
nimcontroller   48000/tcp    # Nimbus Controller
nimspooler      48001/udp    # Nimbus Spooler
nimspooler      48001/tcp    # Nimbus Spooler
nimhub          48002/udp    # Nimbus Hub
nimhub          48002/tcp    # Nimbus Hub
nimgtw          48003/udp    # Nimbus Gateway
nimgtw          48003/tcp    # Nimbus Gateway
#                           Carstein Seeberg <case@nimsoft.no>
#               48004-48555 Unassigned
isnetserv       48128/tcp   # Image Systems Network Services
isnetserv       48128/udp   # Image Systems Network Services
blp5            48129/tcp   # Bloomberg locator
blp5            48129/udp   # Bloomberg locator
#               48130-48555 Unassigned
com-bardac-dw   48556/udp    # com-bardac-dw
com-bardac-dw   48556/tcp    # com-bardac-dw
#                           Nicholas J Howes <nick@ghostwood.org>
#               48557-49150 Unassigned
#               49151       IANA Reserved
```



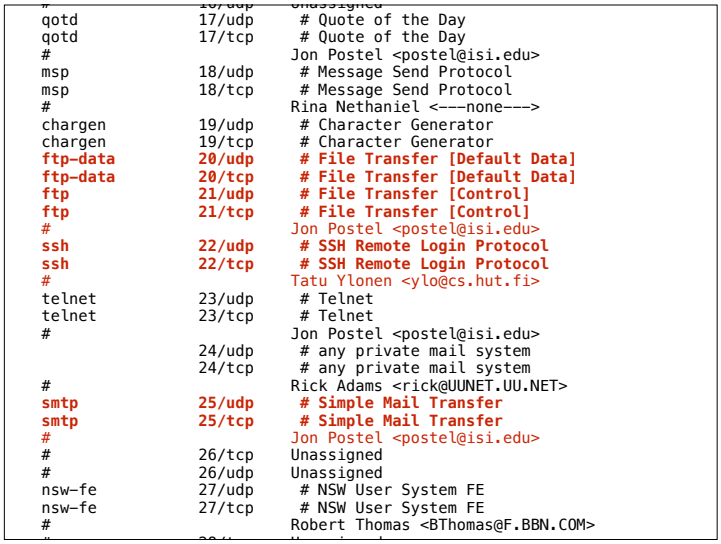
```
discard          9/tcp      # Discard
#                           Jon Postel <postel@isi.edu>
#               10/tcp      Unassigned
#               10/udp      Unassigned
systat          11/udp      # Active Users
systat          11/tcp      # Active Users
#                           Jon Postel <postel@isi.edu>
#               12/tcp      Unassigned
#               12/udp      Unassigned
daytime         13/udp      # Daytime (RFC 867)
daytime         13/tcp      # Daytime (RFC 867)
#                           Jon Postel <postel@isi.edu>
#               14/tcp      Unassigned
#               14/udp      Unassigned
#               15/tcp      Unassigned [was netstat]
#               15/udp      Unassigned
#               16/tcp      Unassigned
#               16/udp      Unassigned
qotd            17/udp      # Quote of the Day
qotd            17/tcp      # Quote of the Day
#                           Jon Postel <postel@isi.edu>
msp             18/udp      # Message Send Protocol
msp             18/tcp      # Message Send Protocol
#                           Rina Nethaniel <---none--->
chargen         19/udp      # Character Generator
chargen         19/tcp      # Character Generator
ftp-data        20/udp      # File Transfer [Default Data]
ftp-data        20/tcp      # File Transfer [Default Data]
ftp             21/udp      # File Transfer [Control]
ftp             21/tcp      # File Transfer [Control]
#                           Jon Postel <postel@isi.edu>
```

FTP = File Transfer Protocol → zum Übertragen von Dateien



```
systat          11/tcp      # Active Users
#                           Jon Postel <postel@isi.edu>
#               12/tcp      Unassigned
#               12/udp      Unassigned
daytime         13/udp      # Daytime (RFC 867)
daytime         13/tcp      # Daytime (RFC 867)
#                           Jon Postel <postel@isi.edu>
#               14/tcp      Unassigned
#               14/udp      Unassigned
#               15/tcp      Unassigned [was netstat]
#               15/udp      Unassigned
#               16/tcp      Unassigned
#               16/udp      Unassigned
qotd            17/udp      # Quote of the Day
qotd            17/tcp      # Quote of the Day
#                           Jon Postel <postel@isi.edu>
msp             18/udp      # Message Send Protocol
msp             18/tcp      # Message Send Protocol
#                           Rina Nethaniel <---none--->
chargen         19/udp      # Character Generator
chargen         19/tcp      # Character Generator
ftp-data        20/udp      # File Transfer [Default Data]
ftp-data        20/tcp      # File Transfer [Default Data]
ftp             21/udp      # File Transfer [Control]
ftp             21/tcp      # File Transfer [Control]
#                           Jon Postel <postel@isi.edu>
ssh             22/udp      # SSH Remote Login Protocol
ssh             22/tcp      # SSH Remote Login Protocol
#                           Tatu Ylonen <ylo@cs.hut.fi>
telnet          23/udp      # Telnet
telnet          23/tcp      # Telnet
```

SSH = Secure Shell → zum Einwählen in andere Rechner



```
qotd            17/udp      # Quote of the Day
qotd            17/tcp      # Quote of the Day
#                           Jon Postel <postel@isi.edu>
msp             18/udp      # Message Send Protocol
msp             18/tcp      # Message Send Protocol
#                           Rina Nethaniel <---none--->
chargen         19/udp      # Character Generator
chargen         19/tcp      # Character Generator
ftp-data        20/udp      # File Transfer [Default Data]
ftp-data        20/tcp      # File Transfer [Default Data]
ftp             21/udp      # File Transfer [Control]
ftp             21/tcp      # File Transfer [Control]
#                           Jon Postel <postel@isi.edu>
ssh             22/udp      # SSH Remote Login Protocol
ssh             22/tcp      # SSH Remote Login Protocol
#                           Tatu Ylonen <ylo@cs.hut.fi>
telnet          23/udp      # Telnet
telnet          23/tcp      # Telnet
#                           Jon Postel <postel@isi.edu>
                24/udp      # any private mail system
                24/tcp      # any private mail system
#                           Rick Adams <rick@UUNET.UU.NET>
smtp            25/udp      # Simple Mail Transfer
smtp            25/tcp      # Simple Mail Transfer
#                           Jon Postel <postel@isi.edu>
#               26/tcp      Unassigned
#               26/udp      Unassigned
nsw-fe          27/udp      # NSW User System FE
nsw-fe          27/tcp      # NSW User System FE
#                           Robert Thomas <BThomas@F.BBN.COM>
```

SMTP = Simple Mail Transfer Protocol → liefert e-mail aus

```
netrjs-4        74/udp     # Remote Job Service
netrjs-4        74/tcp     # Remote Job Service
#                          Bob Braden <Braden@ISI.EDU>
                75/udp     # any private dial out service
                75/tcp     # any private dial out service
#                          Jon Postel <postel@isi.edu>
deos            76/udp     # Distributed External Object Store
deos            76/tcp     # Distributed External Object Store
#                          Robert Ullmann <ariel@world.std.com>
                77/udp     # any private RJE service
                77/tcp     # any private RJE service
#                          Jon Postel <postel@isi.edu>
vettcp          78/udp     # vettcp
vettcp          78/tcp     # vettcp
#                          Christopher Leong <leong@kolmod.mlo.d
finger          79/udp     # Finger
finger          79/tcp     # Finger
#                          David Zimmerman <dpz@RUTGERS.EDU>
http            80/udp      www www-http # World Wide Web HTTP
http            80/tcp      www www-http # World Wide Web HTTP
#                          Tim Berners-Lee <timbl@W3.org>
hosts2-ns       81/udp     # HOSTS2 Name Server
hosts2-ns       81/tcp     # HOSTS2 Name Server
#                          Earl Killian <EAK@MORDOR.S1.GOV>
xfer            82/udp     # XFER Utility
xfer            82/tcp     # XFER Utility
#                          Thomas M. Smith <Thomas.M.Smith@lmco.com>
mit-ml-dev      83/udp     # MIT ML Device
mit-ml-dev      83/tcp     # MIT ML Device
#                          David Reed <--none--->
ctf             84/udp     # Common Trace Facility
```

HTTP = HyperText Transfer Protocol → Liefert Webseiten aus

Ethernet-Server

- Arduino stellt *Serverbibliothek* zur Verfügung
- Wird mit "EthernetServer server(port)" angelegt
- "Ethernet.begin(mac, ip)" startet die Netzverbindung
- "server.begin()" startet den Server

```
EthernetServer server(12345);
Ethernet.begin(mac, ip);
server.begin();
```

DHCP

- Die meisten Geräte lassen sich ihre IP-Adresse über den DHCP-Dienst *zuweisen*
- Wird von (fast) allen Routern unterstützt
- "Ethernet.begin(mac)" weist IP automatisch zu
- Liefert 0 bei Fehlschlagen, 1 bei Erfolg

```
EthernetServer server(12345);
if (Ethernet.begin(mac) == 0)
    Ethernet.begin(mac, ip);
```

Wie anschließen?

Drei Möglichkeiten:

1. Arduino-Board an Router (empfohlen)
2. Arduino-Board an Rechner (mit Internetfreigabe)
3. Arduino-Board an Rechner (ohne Internetfreigabe)

Arduino an Router

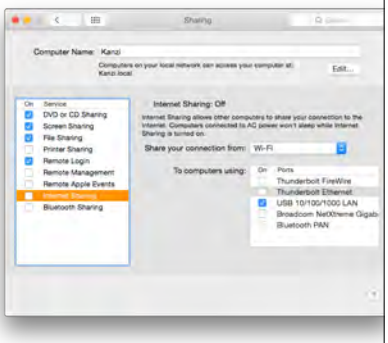
- Ethernet-Kabel in Router stecken
- Arduino erhält IP-Adresse über DHCP

a) am Router (einfach; empfohlen)
* Ethernet-Shield mit Router verbinden
* Arduino erhaelt IP-Adresse ueber DHCP

Arduino an Rechner

– *mit* Internetfreigabe –

- Ethernet-Kabel in Rechner stecken
- Am Rechner *Internetfreigabe* aktivieren
- Arduino erhält IP-Adresse über DHCP



b) am Computer mit Internet Sharing
* Ethernet-Shield mit Rechner verbinden
* Am Computer Internetverbindungsfreigabe ("Internet Connection Sharing") aktivieren, so dass der Internet-Anschluss (WLAN?) ueber den Ethernet-Anschluss geteilt wird.
* Arduino erhaelt IP-Adresse ueber DHCP

Arduino an Rechner

– *ohne* Internetfreigabe –

- Ethernet-Kabel in Rechner stecken
- Ethernet-Anschluss Adresse explizit zuweisen

c) am Computer direkt (schwer)
* Ethernet-Shield mit Rechner verbinden
* Den Ethernet-Anschluss am Computer wie folgt konfigurieren:
  IP-Adresse: 192.168.0.1
  Subnetz-Maske: 255.255.255.0
  Gateway: keins
  Router: keiner
* Arduino sollte ueber die im Programm codierte Adresse ansprechbar sein

---

Server starten

```
// MAC-Adresse
byte mac[] = {
  0xAF, 0xFE, 0xAB, 0xBA, 0xDE, 0xAF
};

// IP-Adresse
IPAddress ip(192, 168, 0, 42);

// Server
EthernetServer server(12345);

void setup() {
  // Serielle Schnittstelle zur Ausgabe der IP nutzen
  Serial.begin(9600);

  // Ethernet-Verbindung und Server starten
  if (Ethernet.begin(mac) == 0)  // DHCP nutzen
  {
      Serial.println("DHCP-Anfrage fehlgeschlagen");
      Ethernet.begin(mac, ip);   // Adresse manuell festlegen
  }

  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
```

Ein Echo-Dienst

- "EthernetClient client = server.available()" liefert verfügbaren Client zurück
- "client.read()" liest ein Zeichen vom Client
- "client.print(s)" schickt eine Zeichenkette s

```
// Echo-Dienst
EthernetClient client = server.available();
if (client) {
    char c = client.read();
    char s[2]; s[0] = c; s[1] = '\0';
    client.print(s);
}
```

Alternativ: "client.write(c)" – schreibt ein einzelnes Zeichen c

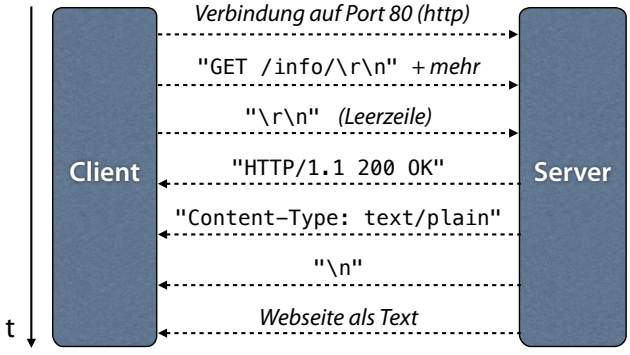
Demo

Telnet-Verbindung zu Port 12345 aufbauen – alles wird zurückgegeben

Web

- Ein *Webserver* (= ein Rechner) wartet auf Port 80, dass sich ein *Webclient* (= noch ein Rechner) mit ihm verbindet.
- Der Client sendet eine *Anfrage* nach einer bestimmten Webseite
- Der Server liefert diese Webseite aus

http://192.168.0.42/info/

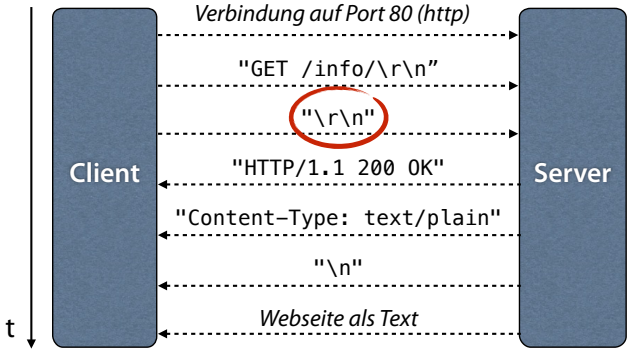


Server starten

```
// MAC-Adresse
byte mac[] = {
  0xAF, 0xFE, 0xAB, 0xBA, 0xDE, 0xAF
};

// IP-Adresse
IPAddress ip(192, 168, 0, 42);

// Server
EthernetServer server(80);

void setup() {
  // Serielle Schnittstelle zur Ausgabe der IP nutzen
  Serial.begin(9600);

  // Ethernet-Verbindung und Server starten
  // Ethernet-Verbindung und Server starten
  if (Ethernet.begin(mac) == 0)  // DHCP nutzen
  {
      Serial.println("DHCP-Anfrage fehlgeschlagen");
      Ethernet.begin(mac, ip);   // Adresse manuell festlegen
  }
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
```

Neu: Jetzt auf Port 80

http://192.168.0.42/info/



Warten auf Leerzeile

- Neben dem Befehl (GET) sendet der Browser noch Informationen über sich
- Wir lesen, bis wir eine Leerzeile gesehen haben
- Eine Leerzeile besteht aus zwei aufeinanderfolgenden '\n' (Neue Zeile-Zeichen)
- Dazwischen können auch '\r' (Wagenrücklauf-Zeichen) stehen

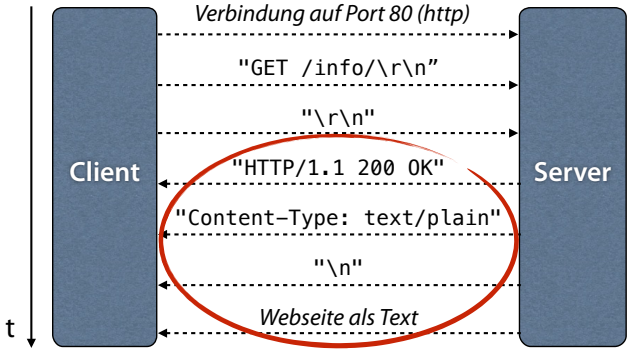
```
void loop() {
  // Anfrage bearbeiten
  EthernetClient client = server.available();
  if (client) {
    int currentLineIsBlank = 1;
    while (client.connected()) {  // Client verbunden
      if (client.available()) {   // Zeichen verfügbar
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          // Antwort schicken...
          break;  // Schleife verlassen
        }
        if (c == '\n') {
          currentLineIsBlank = 1;
        }
        else if (c != '\r') {
          currentLineIsBlank = 0;
        }
      }
    }
    delay(1);
    client.stop();
  }
}
```

Demo

Im Browser Adresse http://192.168.0.42/info/ eingeben – auf serieller Ausgabe sehen, was ankommt

---

http://192.168.0.42/info/



Jetzt müssen wir uns noch um die Antwort kümmern

```
// Anfrage bearbeiten
EthernetClient client = server.available();
if (client) {
  int currentLineIsBlank = 1;
  while (client.connected()) {  // Client verbunden
    if (client.available()) {   // Zeichen verfügbar
      char c = client.read();
      Serial.write(c);
      if (c == '\n' && currentLineIsBlank) {
        // Antwort schicken
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/plain");
        client.println("Connection: close");
        client.println("Refresh: 5");
        client.println();
        client.println("Hello, world!”);
        break;
      }
      if (c == '\n') {
        currentLineIsBlank = 1;
      }
      else if (c != '\r') {
        currentLineIsBlank = 0;
      }
    }
  }
```

Demo

Im Browser Adresse eingeben; jetzt muss “Hello, world” zurückkommen

---

Sensoren lesen

- Wir geben die aktuellen Sensor-Werte aus

```
if (client.available()) {   // Zeichen verfügbar
  char c = client.read();
  Serial.write(c);
  if (c == '\n' && currentLineIsBlank) {
    // Antwort schicken
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/plain");
    client.println("Connection: close");
    client.println("Refresh: 5");
    client.println();

    // Analoge Anschluesse ausgeben
    for (int pin = 0; pin < 6; pin++) {
      int sensorValue = analogRead(pin);
      client.print("Analog-Eingang ");
      client.print(pin);
      client.print(" hat den Wert ");
      client.print(sensorValue);
      client.println();
    }
    break;
  }
  if (c == '\n') {
    currentLineIsBlank = 1;
  }
  else if (c != '\r') {
```

Demo

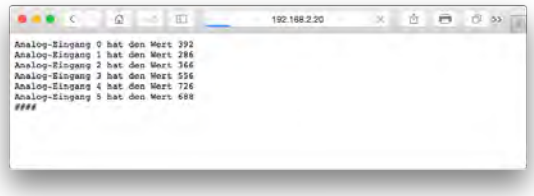
Im Browser Adresse eingeben; jetzt müssen die Werte angezeigt werden

---

Balken

```
// Anschluss 0 als Balken ausgeben
int value = analogRead(0);
for (int i = 0; i < value; i += 100)
  client.print("#");
client.println();
```

Werte ausgeben



Automatische Aktualisierung alle 5 Sekunden

Demo

Im Browser Adresse eingeben; jetzt müssen die Werte angezeigt werden

---

HTML

- Auszeichnungssprache für Webseiten
- Nutzt *Tags* (in <…>) für spezielle Funktionen (Struktur, Format, Interaktion)
- Alles, was nicht in Tags steht, wird als Text dargestellt

Ein HTML-Dokument

```
<!DOCTYPE HTML>
<html>
Willkommen bei Arduino!
</html>
```

HTML-Tags

```
<!DOCTYPE HTML>
<h1>Ein Titel</h1>
<p>Ein Absatz mit
<em>hervorgehobenem Text</em>
und <br/>
<strong>stark hervorgehobenem
Text</strong>.
</p>
<p>Eine ungeordnete Liste:
<ul>
<li> Ein
Aufz&auml;hlungselement </li>
<li> Noch eins </li>
</ul>
und eine geordnete Liste:
<ol>
<li> Nummer eins </li>
<li> Nummer zwei </li>
</ol>
</p>
```

Ein Titel

Ein Absatz mit *hervorgehobenem Text* und **stark hervorgehobenem Text**.
Eine ungeordnete Liste:

- Ein Aufzählungselement
- Noch eins

und eine geordnete Liste:

1. Nummer eins
2. Nummer zwei

```
if (c == '\n' && currentLineIsBlank) {
  // Antwort schicken
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println("Refresh: 5");
  client.println();

  client.println("<!DOCTYPE HTML>");
  client.println("<html>");

  client.println("<h1>Mein Arduino</h1>");

  // Analoge Anschluesse ausgeben
  for (int pin = 0; pin < 6; pin++) {
    int sensorValue = analogRead(pin);
    client.print("Analog-Eingang ");
    client.print(pin);
    client.print(" hat den Wert <strong>");
    client.print(sensorValue);
    client.println("</strong><br />");
  }
  client.println("/<html>");
  break;
}
if (c == '\n') {
```

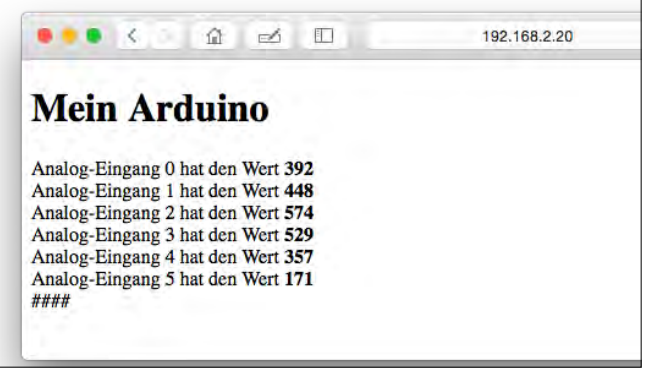
Wir ändern den Inhaltstyp von text/plain (gewöhnlicher Text) in text/html (HTML-Dokument) und fügen passende HTML-Tags ein

Demo

Im Browser Adresse eingeben; jetzt müssen die Werte angezeigt werden

---

Werte mit HTML



Eingaben

- Idee: per Webseite LED kontrollieren
- Mit http://192.168.0.42/on/ – einschalten
- Mit http://192.168.0.42/off/ – ausschalten

Kommando speichern

```
// Puffer fuer das empfangene Kommando
const int COMMAND_SIZE = 2048;
char command[COMMAND_SIZE];


int cmd = 0;
while (client.connected()) {  // Client verbunden
  if (client.available()) {   // Zeichen verfuegbar
    char c = client.read();
    Serial.write(c);
    if (cmd < COMMAND_SIZE - 1)
      command[cmd++] = c;
    if (c == '\n' && currentLineIsBlank) {
      command[cmd] = '\0';
```

“const” = Der Wert ändert sich nicht; nützlich für Feldgrößen

---

Kommando befolgen

```
const int led = 13;

void setup() {
  ...
  // LED schalten
  pinMode(led, OUTPUT);
}

void loop () {
  ...
  // Auf Kommando reagieren
  if (strncmp(command, "GET /on", 7) == 0) {
    digitalWrite(led, HIGH);
  }
  if (strncmp(command, "GET /off", 8) == 0) {
    digitalWrite(led, LOW);
  }
```

“const” = Der Wert ändert sich nicht; nützlich für Feldgrößen
strncmp(s1, s2, n) – n erste Zeichen der Zeichenketten s1 und s2 vergleichen (hier 7 bzw. 8)

Demo

Verweise

- In HTML kann mit
  `<a href="URL">Text</a>`
  auf andere Webseiten verwiesen werden
- URLs ohne Hostnamen (www.foo.com)
  verweisen auf aktuellen Host

Verweise ausgeben

```
client.println("<p>");
client.println("LED <a href=\"/on\">einschalten</a>");
client.println(" | ");
client.println("<a href=\"/off\">ausschalten</a>");
client.println("</p>");
```

erzeugt

```
<p>
LED <a href="/on">einschalten</a>
 |
<a href="/off">ausschalten</a>
</p>
```

\" = Anführungszeichen innerhalb einer Zeichenkette

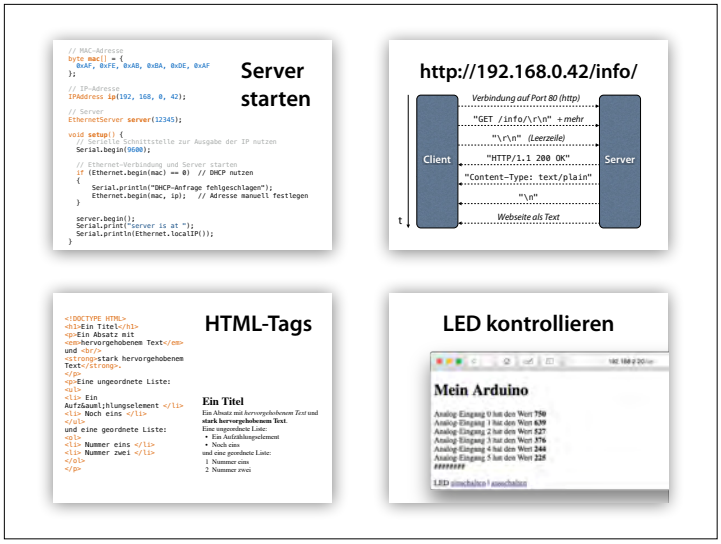
Demo

LED kontrollieren

Zugangskontrolle

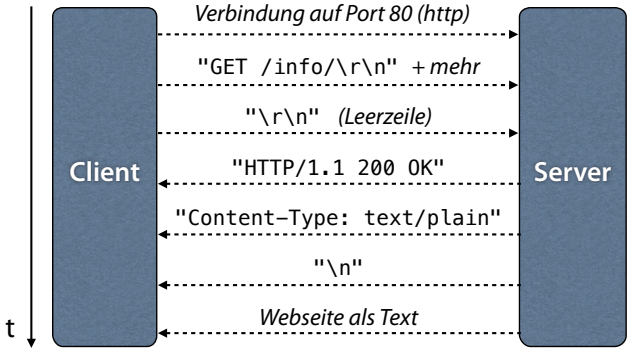
- Hinter einem Router oder Rechner ist Ihr Arduino nicht für das Internet sichtbar
- Vom Internet aus kann *jeder* auf Ihr Gerät zugreifen (und "geheime" URLs mitschneiden)
- Bevor Sie Ihr Programm ins Internet stellen, *wenden Sie sich an Ihren freundlichen Informatiker*

HTTP
MAC
SHIELD
DHCP
HTML
FREIGABE
PORT
IP
ROUTER
ETHERNET
INTERNET



Handouts

http://192.168.0.42/info/



Server starten

```
// MAC-Adresse
byte mac[] = {
  0xAF, 0xFE, 0xAB, 0xBA, 0xDE, 0xAF
};

// IP-Adresse
IPAddress ip(192, 168, 0, 42);

// Server
EthernetServer server(80);

void setup() {
  // Serielle Schnittstelle zur Ausgabe der IP nutzen
  Serial.begin(9600);

  // Ethernet-Verbindung und Server starten
  // Ethernet-Verbindung und Server starten
  if (Ethernet.begin(mac) == 0)  // DHCP nutzen
  {
      Serial.println("DHCP-Anfrage fehlgeschlagen");
      Ethernet.begin(mac, ip);   // Adresse manuell festlegen
  }
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
```

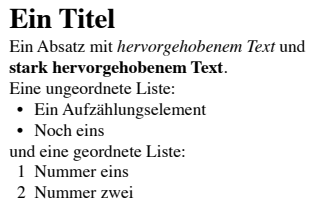
Neu: Jetzt auf Port 80

```
void loop() {
  // Anfrage bearbeiten
  EthernetClient client = server.available();
  if (client) {
    int currentLineIsBlank = 1;
    while (client.connected()) {  // Client verbunden
      if (client.available()) {   // Zeichen verfügbar
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          // Antwort schicken...
          break;  // Schleife verlassen
        }
        if (c == '\n') {
          currentLineIsBlank = 1;
        }
        else if (c != '\r') {
          currentLineIsBlank = 0;
        }
      }
    }
    delay(1);
    client.stop();
  }
}
```

```
      if (client.available()) {   // Zeichen verfügbar
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          // Antwort schicken
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/plain");
          client.println("Connection: close");
          client.println("Refresh: 5");
          client.println();

          // Analoge Anschluesse ausgeben
          for (int pin = 0; pin < 6; pin++) {
            int sensorValue = analogRead(pin);
            client.print("Analog-Eingang ");
            client.print(pin);
            client.print(" hat den Wert ");
            client.print(sensorValue);
            client.println();
          }
          break;
        }
        if (c == '\n') {
          currentLineIsBlank = 1;
        }
        else if (c != '\r') {
```

HTML-Tags

```
<!DOCTYPE HTML>
<h1>Ein Titel</h1>
<p>Ein Absatz mit
<em>hervorgehobenem Text</em>
und <br/>
<strong>stark hervorgehobenem
Text</strong>.
</p>
<p>Eine ungeordnete Liste:
<ul>
<li> Ein
Aufz&auml;hlungselement </li>
<li> Noch eins </li>
</ul>
und eine geordnete Liste:
<ol>
<li> Nummer eins </li>
<li> Nummer zwei </li>
</ol>
</p>
```



Ein Titel

Ein Absatz mit *hervorgehobenem Text* und **stark hervorgehobenem Text**.
Eine ungeordnete Liste:

- Ein Aufzählungselement
- Noch eins

und eine geordnete Liste:

1. Nummer eins
2. Nummer zwei

Kommando speichern

```
// Puffer fuer das empfangene Kommando
const int COMMAND_SIZE = 2048;
char command[COMMAND_SIZE];

int cmd = 0;
while (client.connected()) {  // Client verbunden
  if (client.available()) {   // Zeichen verfuegbar
    char c = client.read();
    Serial.write(c);
    if (cmd < COMMAND_SIZE - 1)
      command[cmd++] = c;
    if (c == '\n' && currentLineIsBlank) {
      command[cmd] = '\0';
```

"const" = Der Wert ändert sich nicht; nützlich für Feldgrößen

---

Kommando befolgen

```
const int led = 13;

void setup() {
  ...
  // LED schalten
  pinMode(led, OUTPUT);
}

void loop () {
  ...
  // Auf Kommando reagieren
  if (strncmp(command, "GET /on", 7) == 0) {
    digitalWrite(led, HIGH);
  }
  if (strncmp(command, "GET /off", 8) == 0) {
    digitalWrite(led, LOW);
  }
```

"const" = Der Wert ändert sich nicht; nützlich für Feldgrößen
strncmp(s1, s2, n) – n erste Zeichen der Zeichenketten s1 und s2 vergleichen (hier 7 bzw. 8)

Verweise ausgeben

```
client.println("<p>");
client.println("LED <a href=\"/on\">einschalten</a>");
client.println(" | ");
client.println("<a href=\"/off\">ausschalten</a>");
client.println("</p>");
```

erzeugt

```
<p>
LED <a href="/on">einschalten</a>
 |
<a href="/off">ausschalten</a>
</p>
```

\" = Anführungszeichen innerhalb einer Zeichenkette