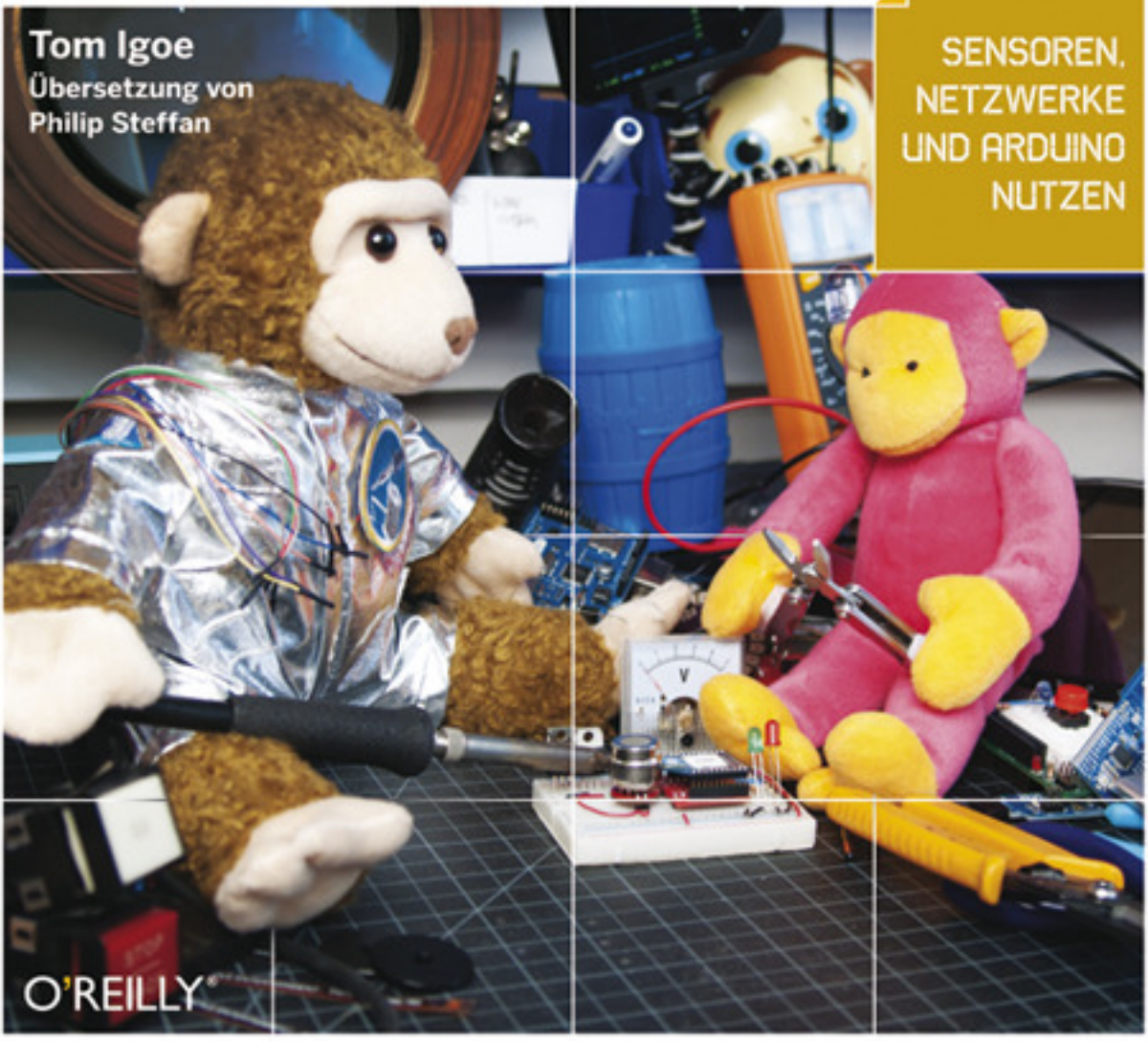
Making Things Talk

Lernen durch Entdecken

Die Welt hören, sehen, fühlen



Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628

Inhalt

Vorwort ..... **vii**
Für wen dieses Buch gedacht ist viii
Was du wissen musst ix
Inhalt dieses Buchs x
Bauteile kaufen xi
Nutzung von Codebeispielen xi
Nutzung der Schaltungsbeispiele xii
Danksagungen für die Erstauflage xii
Hinweis zur zweiten Auflage xv

Kapitel 1: Die Werkzeuge **1**
Es beginnt mit den Dingen, die man anfasst 2
Es geht um Impulse 2
Computer in allen Formen und Größen 3
Gute Angewohnheiten 4
Werkzeuge 5
Nutzung der Befehlszeile 13
Benutzung eines Oszilloskops 34
Es endet mit den Dingen, die man anfasst 35

Kapitel 2: Das einfachste Netzwerk **37**
Materialien für Kapitel 2 38
Einigung in Schichten 40
Herstellen der Verbindung: Die unteren Schichten 42
Projekt 1: Type Brighter 46
Projekt 2: Monski Pong 50
Flusssteuerung 62
Projekt 3: Drahtloses Monski Pong 64
Projekt 4: Verhandeln mit Bluetooth 68
Fazit 72

Kapitel 3: Ein komplexeres Netzwerk **75**
Materialien für Kapitel 3 76
Netzkarten und Adressen 77
Projekt 5: Vernetzte Katze 89
Fazit 112

Kapitel 4: Es geht auch ohne Computer! Mikrocontroller im Internet 115
Materialien für Kapitel 4 117
Einführung: Netzwerkmodule 118
Projekt 6: Hallo, Internet! 120
Eine Embedded-Netzwerk-Client-Anwendung 127
Projekt 7: Internet-Luftqualitätsmesser 127
Tools zum Programmieren und zur Fehlersuche bei Embedded-Modulen 140
Fazit 147

Kapitel 5: Übertragungen in (Fast-)Echtzeit 149
Materialien für Kapitel 5 150
Interaktive Systeme und Feedback-Schleifen 151
Transmission Control Protocol: Sockets & Sessions 152
Projekt 8: Netzwerk-Pong 153
Die Clients 155
Fazit 178

Kapitel 6: Drahtlose Kommunikation 181
Materialien für Kapitel 6 182
Warum ist nicht alles drahtlos? 184
Zwei drahtlose Varianten: Infrarot und Funk 185
Projekt 9: Infrarotsteuerung einer Digitalkamera 188
So funktioniert Funk 190
Projekt 10: Doppelübertragung per Funk 193
Projekt 11: Bluetooth-Transceiver 206
Funkmodule kaufen 216
Was ist mit WLAN 216
Projekt 12: Hallo WLAN! 217
Fazit 220

Kapitel 7: Sessionlose Netzwerke 223
Materialien für Kapitel 7 224
Sessions oder Nachrichten 226
Wer ist da draußen? Broadcast-Nachrichten 227
Projekt 13: Giftwarnung in der Werkstatt 232
Adressierte Nachrichten 246
Projekt 14: Solarzellen-Daten drahtlos weiterleiten 248
Fazit 258

Kapitel 8: Wie man (fast) alles orten kann 261
Materialien für Kapitel 8 262
Ortung im Netzwerk und im physischen Raum 264
Entfernung bestimmen 267
Projekt 15: Beispiele zur Infrarot-Entfernungsmessung 268
Projekt 16: Beispiele zur Ultraschall-Entfernungsmessung 270
Projekt 17: Auslesen der Signalstärke mit XBee-Funkmodulen 273
Projekt 18: Auslesen der Signalstärke mit Bluetooth-Funkmodulen 276
Positionsbestimmung durch Trilateration 277
Projekt 19: Auslesen des GPS-Protokolls 278

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Bestimmung der Himmelsrichtung 286
Projekt 20: Richtungsbestimmung mit einem Digitalkompass 286
Projekt 21: Lagebestimmung mit einem Beschleunigungsmesser 290
Fazit 299

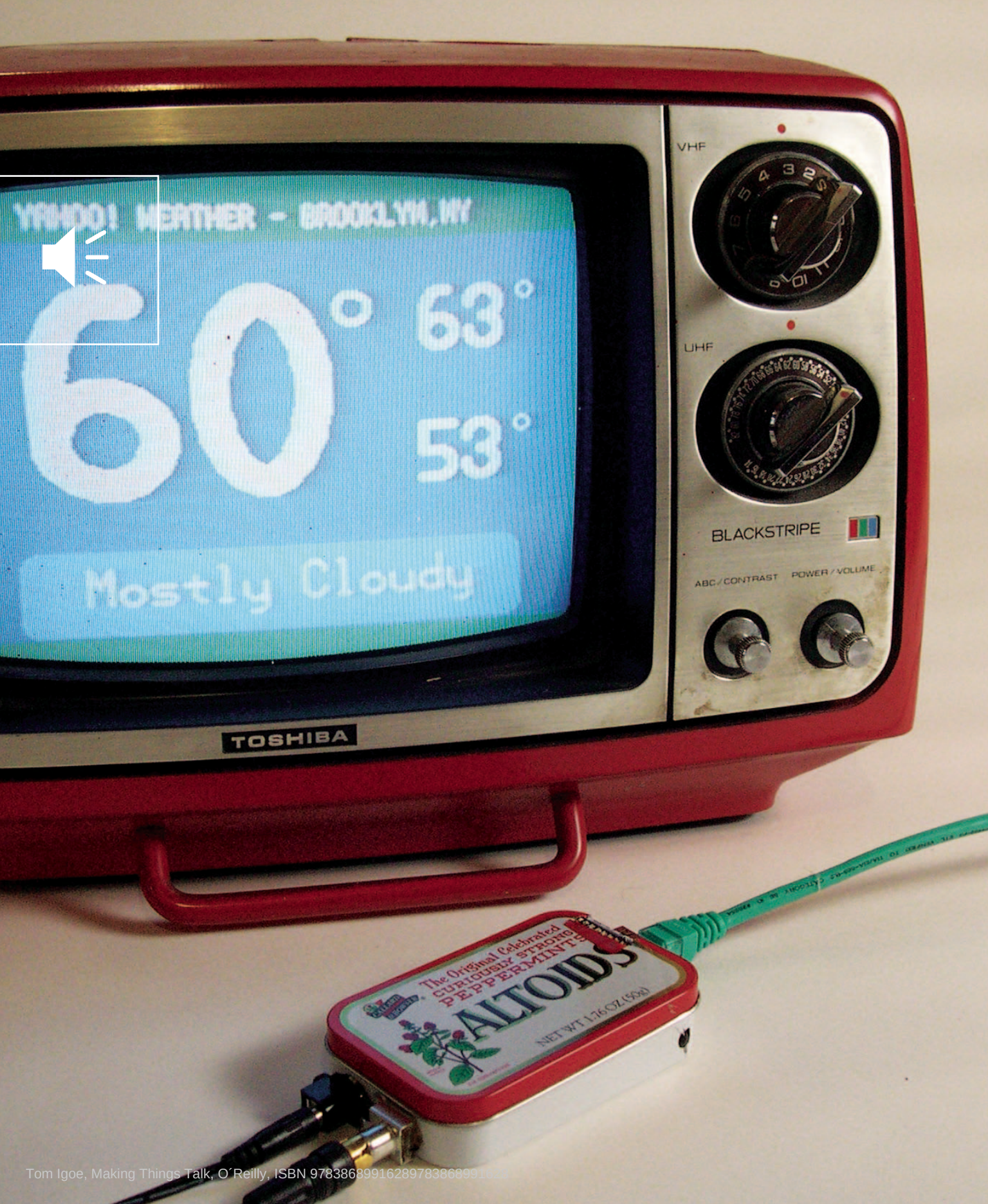
Kapitel 9: Identifizierung 301
Materialien für Kapitel 9 302
Physische Identifizierung 304
Projekt 22: Farberkennung mit einer Webcam 307
Projekt 23: Gesichtserkennung mit einer Webcam 311
Projekt 24: Erkennung von 2-D-Barcodes mit einer Webcam 314
Projekt 25: Auslesen von RFID-Tags mit Processing 320
Projekt 26: RFID trifft auf Heim-Automation 323
Projekt 27: Tweets mit RFID 331
Netzwerk-Identifizierung 355
Projekt 28: IP-Geokodierung 357
Fazit 362

Kapitel 10: Mobilfunknetze und die physische Welt 365
Materialien für Kapitel 10 366
Ein großes Netz 368
Projekt 29: Katzen-Cam 2 371
Projekt 30: Thermostat am Apparat 388
Textnachrichten-Interfaces 395
Native Anwendungen für Mobiltelefone 398
Projekt 31: Persönlicher mobiler Datenlogger 403
Fazit 417

Kapitel 11: Protokolle unter der Lupe 419
Materialien für Kapitel 11 420
Herstellen der Verbindung 421
Text oder Binär? 424
MIDI 427
Projekt 32: Spaß mit MIDI 429
Representational State Transfer 437
Projekt 33: Spaß mit REST 439
Fazit 442

Anhang: Wie du an das Elektronik-Zeugs kommst 445
Bezugsliste in Deutschland 446
Supplies 447

Index 457



Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628978386899162

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

4

MAKE: PROJECTS

Es geht auch ohne Computer! Mikrocontroller im Internet

Nachdem sie ein Projekt wie das Online-Katzenbett aus Kapitel 3 gebaut haben, fällt den meisten Menschen als Erstes ein: „Toll, aber wie kann ich das machen, ohne dass es an meinen Computer angeschlossen sein muss?“

Es ist umständlich, den Mikrocontroller mit einem Laptop oder PC verbinden zu müssen, nur damit er ins Internet gehen kann.

Du hast ja in Kapitel 3 erfahren, dass Internetprotokolle einfache Textnachrichten sind und dass Mikrocontroller sich gut dafür eignen, ebensolche zu schicken. In diesem Kapitel lernst du daher, wie man einen Mikrocontroller mit dem Internet verbindet, mit einem Bauteil, das nicht viel komplexer als das Bluetooth-Funkmodem aus Kapitel 2 ist.

◀◀ **Die YBox von Uncommon Projects** (http://uncommonprojects.com/site/play/ybox-2) zeigt RSS-Feeds auf einem Fernseher mithilfe eines Seriell-Ethernet-Adapters von XPort und eines Propeller-Mikrochips an. *Bild mit freundlicher Genehmigung von Uncommon Projects.*

In den letzten Jahren ist eine große Bandbreite von kommerziellen Geräten auf den Markt gekommen, die sich direkt ohne die Hilfe eines PCs mit dem Internet verbinden lassen. Hersteller wie D-Link, Sony, Axis und andere bieten Überwachungskameras mit Netzwerkschnittstellen an, sowohl Ethernet als auch WLAN. Cevia, eStarling und andere stellen Bilderrahmen mit WLAN-Fähigkeiten her, auf die man über das Internet Bilder laden kann. Ambient Devices baut Lampen und verschiedene Displays, die ins Netz gehen und ihr Erscheinungsbild anhand von Informationen wie z.B. Börsendaten, dem Wetter und anderen numerischen Größen verändern. Set-Top-Boxen von Kabelanbietern sind Computer in einem kleinen Gehäuse, die gleichzeitig Audio-, Video- und Datenströme verarbeiten können. Das Betriebssystem in deiner Set-Top-Box ist möglicherweise sogar eine Variante des Linux-Betriebssystems, das auch auf dem Serverrechner bei deinem Webhoster läuft.

Alarmanlagensysteme bestehen aus einem Netzwerk von Mikrocontrollern, die miteinander reden und von denen einer mit einem zentralen Server kommuniziert, meist mithilfe eines Modems über die Telefonleitung.

Diese Geräte eint, dass sie Kommunikation in Netzwerken betreiben. Die einfachsten von ihnen können immer nur einen Vorgang durchführen, z.B. Daten bei einem Server anfragen und auf die Antwort warten oder als Reaktion auf ein physisches Ereignis eine E-Mail losschicken. Andere können mehrere Übertragungsströme gleichzeitig regeln, so dass du gleichzeitig ins Internet gehen und fernsehen kannst. Je mehr Rechenleistung ein bestimmtes Gerät hat, desto mehr kann es bearbeiten. Für die meisten Anwendungen brauchst du aber gar keine große Rechenleistung, weil das Gerät, das du baust, nur ein oder zwei Funktionen hat.

X

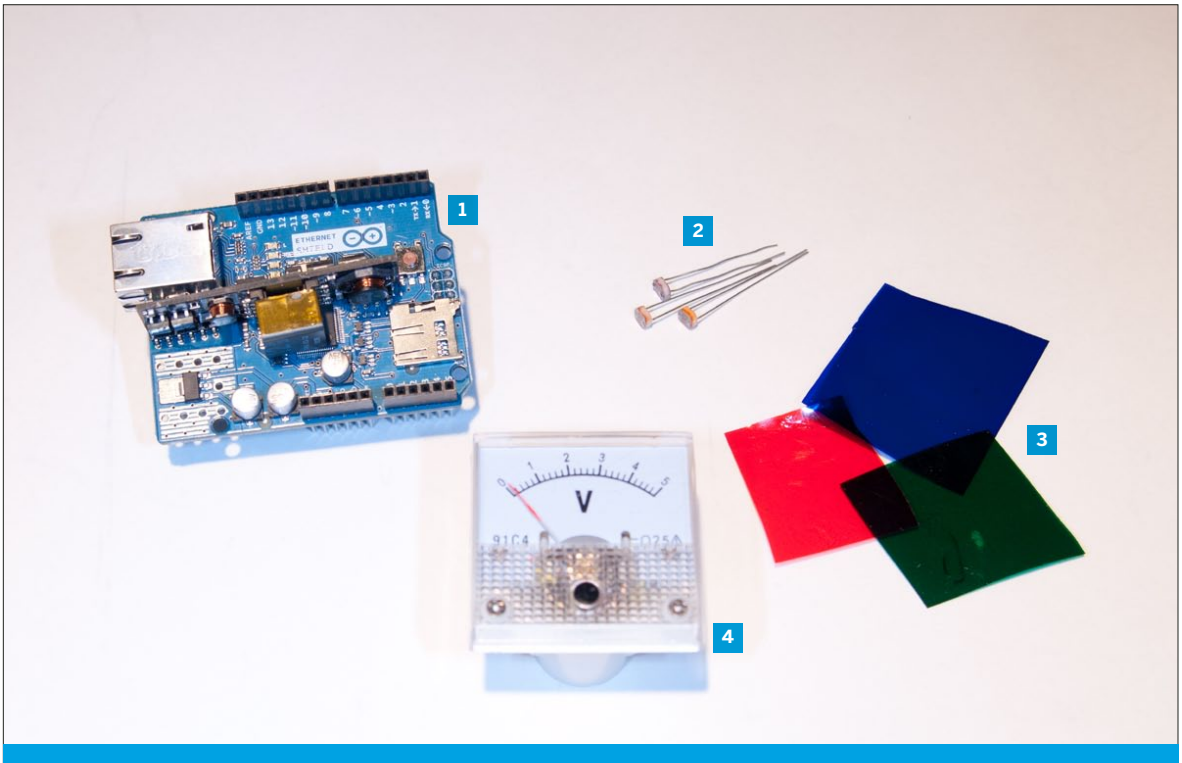


Abbildung 4-1. Neue Teile für dieses Kapitel: **1.** Arduino Ethernet-Shield. Dieses Modell hat das optionale Power-over-Ethernet-Board **2.** Fotozellen **3.** Farbfilter für Lampen in rot, grün und blau **4.** Voltmeter. Du kannst ein neues kaufen, aber ein antikes Modell wäre noch besser. Vergiss nicht, auch einen Vorrat an Steckleisten zur Verbindung der Module zu besorgen.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Materialien für Kapitel 4

HERSTELLER-LEGENDE

- **A** Arduino Store (http://store.arduino.cc/ww)
- **AF** Adafruit (http://adafruit.com)
- **D** Digi-Key (www.digikey.com)
- **F** Farnell (www.farnell.com)
- **J** Jameco (http://jameco.com)
- **MS** Maker SHED (www.makershed.com)
- **RS** RS (www.rs-online.com)
- **SF** SparkFun (www.sparkfun.com)
- **SS** Seeed Studio (www.seeedstudio.com)

PROJEKT 6: Hallo, Internet! Lichtfarben-Webserver

» **Ein Arduino-Ethernet-Modul** **A** A000050
Alternativ geht auch ein Uno-kompatibles Modul (siehe Kapitel 2) mit einem Ethernet-Shield.
SF DEV-09026, **J** 2124242, **A** A000056, **AF** 201, **F** 1848680

» **Eine Ethernet-Verbindung zum Internet** Dein Heimrouter hat sehr wahrscheinlich Ethernet-Buchsen an der Rückseite. Wenn du deinen Computer über ein Kabel mit dem Internet verbunden hast, weißt du, wo die Buchsen sind.

» **3 10-kΩ-Widerstände** **D** 10KQBK-ND, **J** 29911, **F** 9337687, **RS** 707-8906

» **3 Fotozellen (lichtabhängige Widerstände)** **D** PDV-P9200-ND, **J** 202403, **SF** SEN-09088, **F** 7482280, **RS** 234-1050

» **Ein Steckbrett** **D** 438-1045-ND, **J** 20723 oder 20601, **SF** PRT-00137, **F** 4692810, **AF** 64, **SS** STR101C2M oder STR102C2M, **MS** MKKN2

» **3 Farbfolien** In den Primärfarben (primary red, primary green, primary blue). Erhältlich beim lokalen Lichtequipment- oder Fotozubehör-Händler.

PROJEKT 7: Internet-Luftqualitätsmesser

» **Ein Arduino-Ethernet-Modul** **A** A000050
Alternativ geht auch ein Uno-kompatibles Modul (siehe Kapitel 2) mit einem Ethernet-Shield.
SF DEV-09026, **J** 2124242, **A** A000056, **AF** 201, **F** 1848680

» **Eine Ethernet-Verbindung zum Internet** Dein Heimrouter hat sehr wahrscheinlich Ethernet-Buchsen an der Rückseite. Wenn du deinen Computer über ein Kabel mit dem Internet verbunden hast, weißt du, wo die Buchsen sind.

» **Ein Steckboard** **D** 438-1045-ND, **J** 20723 oder 20601, **SF** PRT-00137, **F** 4692810, **AF** 64, **SS** STR101C2M oder STR102C2M, **MS** MKKN2

» **Ein Voltmeter** Versuch, ein schickes antikes Modell zu bekommen. Idealerweise sollte es im Bereich von 0–5 oder maximal 0–10 Volt messen.
SF TOL-10285, **F** 4692810, **RS** 244-890

» **4 LEDs** **D** 160-1144-ND oder 160-1665-ND, **J** 34761 oder 94511, **F** 1015878, **RS** 247-1662 oder 826-830, **SF** COM-09592 oder COM-09590

» **4 220-Ohm-Widerstände** **D** 220QBK-ND, **J** 690700, **F** 9337792, **RS** 707-8842

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Einführung: Netzwerkmodule

Es ist möglich, ein Programm für einen Mikrocontroller zu schreiben, das alle Schritte der Netzwerkkommunikation durchführen kann, von der Herstellung der Verbindung über das Management der Netzwerkadresse bis zur Verhandlung in Protokollen wie SMTP und HTTP. Eine Codebibliothek, die alle nötigen Schichten für Netzkommunikation umfasst, nennt man einen Netzwerk-Stack oder TCP/IP-Stack. Es ist aber viel einfacher, für diese Aufgabe ein Netzwerkmodul zu benutzen.

Im Handel sind viele Module erhältlich, die sich in Preis und Fähigkeiten unterscheiden Die Einstiegshürden und das nötige technische Verständnis sind bei verschiedenen Mikrocontroller-Plattformen unterschiedlich. Das ist bei Netzwerk-Controllern genau so. Einige Module, z.B. RabbitCore-Prozessoren von Rabbit Semiconductor, werden mit dem Quelltext eines TCP/IP-Stacks geliefert, den du deinen Bedürfnissen anpassen musst, um und dann das Gerät selbst zu programmieren. Andere wie das Beagle Board sind vollständige netzwerkfähige Computer auf einer einzelnen Platine. Sie sind sehr leistungsfähig und bieten erfahrenen Netzwerkprogrammierern eine sehr komfortable Programmierungsumgebung. Allerdings ist es dort nicht so leicht, Sensoren und Aktoren anzuschließen. Die Lernkurve ist für Menschen ohne Erfahrung in Netzwerkprogrammierung sehr steil. Andere wie die Module von Lantronix (XPort, XPort Direct, MatchPort und WiPort) haben einen in der Firmware einprogrammierten Stack und bieten dir ein serielles, telnet- oder webbasiertes Interface. Sie sind viel einfacher zu benutzen. Das Webinterface ermöglicht dir den Zugriff über den Browser auf deinem PC, das telnet-Interface ermöglicht dir den Zugriff von einem Server oder deinem PC aus und das serielle Interface ermöglicht dir den Zugriff über einen Mikrocontroller. Das sind Seriell-zu-Ethernet-Modems. Sie funktionieren ähnlich wie die Bluetooth-Modems, die du in Kapitel 2 benutzt hast, aber sie haben ein anderes serielles Protokoll. Sie eignen sich gut für den Einstieg und ich habe sie in der ersten Auflage dieses Buchs ständig eingesetzt. In dieser Neuauflage lernst du allerdings Ethernet-Module mit Synchron-serieller Schnittstelle kennen. Diese sind so einfach zu benutzen wie die Seriell-zu-Ethernet-Module, belegen aber keinen asynchronen seriellen Port an deinem Mikrocontroller. Das bedeutet, dass du den seriellen Port

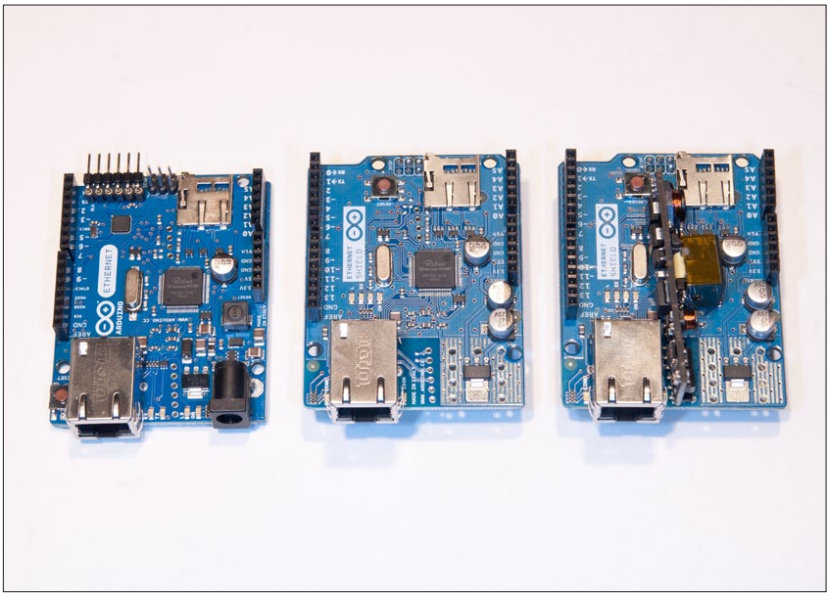


Abbildung 4-2
Der Arduino Ethernet (links), ein Arduino Ethernet-Shield (Mitte) und ein Ethernet-Shield mit Power-over-Ethernet-Modul (rechts). Alle sind für die Ethernet-Projekte in diesem Buch geeignet.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

noch zum Debuggen oder zur Kommunikation mit anderen Geräten benutzen kannst.

Es gibt beim Arduino zwei Möglichkeiten für eine Ethernetverbindung. Die erste ist das Arduino-Ethernet-Modul, das ein Arduino-Modul ist, das direkt auf der Platine einen Ethernet-Baustein hat. Der Ethernet-Anschluss ersetzt den USB-Anschluss, den Standard-Arduinos haben. Um dieses Modul zu programmieren, brauchst du zusätzlich einen USB-zu-seriell-Adapter nach FTDI-Standard. Die andere Möglichkeit ist eine Zusatzplatine für normale Arduino-Module namens Ethernet Shield. Davon sind einige Versionen auf dem Markt. Der Arduino-Ethernet-Shield setzt einen Ethernet-Chip von WizNet ein, den W5100. Dieser Shield verfügt außerdem über einen Steckplatz für eine SD-Karte, genau wie das Arduino-Ethernet-Modul. Der Ethernet-Shield von Adafruit nutzt ebenfalls den W5100-Chip, kann aber auch das eingangs erwähnte Seriell-zu-Ethernet-Modul von Lantronix nutzen. Die Projekte in diesem Kapitel funktionieren mit dem Arduino Ethernet, mit dem Arduino Ethernet-Shield oder mit dem Adafruit Ethernet-Shield mit W5100-Modul. Diese sind in Abbildung 4-2 zu sehen. Die Projekte laufen allerdings nicht mit dem Lantronix-Modul.

X

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Einführung: Serial Peripheral Interface (SPI)

Der Ethernet-Shield und der Arduino Ethernet kommunizieren mit ihren Ethernet-Controllern über eine Methode zur synchron-seriellen Übertragung namens Serial Peripheral Interface, kurz SPI. SPI sowie ein anderes synchron-serielles Protokoll, Inter-Integrated Circuit oder I2C (auch manchmal als Two-Wire Interface, oder TWI bezeichnet), sind die zwei am weitesten verbreiteten synchron-seriellen Protokolle, die dir begegnen werden.

Bei allen synchron-seriellen Protokollen gibt es ein Steuergerät, bei dem ein gleichmäßiger Impuls, das Taktsignal, an einem Pin generiert wird. Bei jedem Taktimpuls werden Daten über einen anderen Pin ausgetauscht (siehe Kapitel 2). Der Vorteil eines synchron-seriellen Protokolls ist, dass es sich um einen Bus handelt: Mehrere Geräte können sich dieselbe physische Verbindung mit einem Hauptcontroller teilen. Jedes Protokoll implementiert den Bus etwas anders.

SPI-Verbindungen bestehen aus drei oder vier Leitungen zwischen dem Steuergerät (dem Master) und den Zusatzgeräten (oder Slaves), und zwar denen hier:

Clock: Der Pin, an dem der regelmäßige Impuls vom Master anliegt.

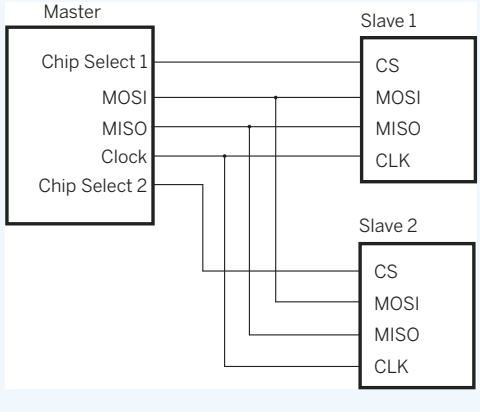
Master Out, Slave In (MOSI): Das Master-Gerät sendet bei jedem Taktimpuls ein Datenbit auf dieser Leitung an den Slave.

Master In, Slave Out (MISO): Das Slave-Gerät sendet bei jedem Taktimpuls ein Datenbit auf dieser Leitung an den Master.

Slave Select (SS) oder Chip Select (CS): Weil sich mehrere Slave-Geräte denselben Bus teilen können, hat jedes eine eigene Verbindung zum Master. Der Master setzt diesen Pin auf low, um genau dieses Gerät anzusprechen. Wenn der Master nicht mit einem bestimmten Slave spricht, setzt er diesen Pin auf high.

Wenn der Slave keine Daten an den Master senden muss, gibt es keinen MISO-Pin.

Die Arduino-SPI-Library benutzt Pin 11 für MOSI, Pin 12 für MISO und Pin 13 für Clock (Takt). Pin 10 ist standardmäßig der Pin für Chip Select, aber wie du noch sehen wirst, kannst du das auch ändern. Der Arduino Ethernet und der Ethernet-Shield haben z.B. zwei Slave-Geräte an ihrem SPI-Bus: Den WizNet-Chip und die SD-Karte. Beide sind mit den Pins 11, 12 und 13 für MISO, MOSI und Clock verbunden. The WizNet-Modul benutzt Pin 10 für Chip Select, die SD-Karte nutzt dagegen Pin 4 für Chip Select.



Projekt 6

Hallo, Internet!

Um ein Netzwerkmodul zu benutzen, musst du es erst einmal mit dem Netzwerk verbinden. Das ist jetzt unser Ziel. Für dieses Projekt baust du einen sehr einfachen Webserver auf deinem Arduino, der eine Webseite ausliefert, deren Hintergrundfarbe sich mit der Lichtfarbe des Ortes ändert, an dem der Arduino liegt.

MATERIALIEN

- **Ein Arduino Ethernet oder**
- **Ein Arduino Ethernet-Shield und ein Arduino-Mikrocontroller-Modul**
- **Eine Ethernet-Verbindung zum Internet**
- **3 10-kΩ-Widerstände**
- **3 Photozellen (lichtabhängige Widerstände)**
- **Ein Steckboard**
- **3 Farbfolien**

Herstellen der Verbindung

Wenn du das Arduino-Ethernet-Modul benutzt, steck es einfach an einen USB-zu-seriell-Adapter, dann taucht es wie ein normaler Arduino in der Liste der seriellen Ports auf. Der USB-zu-seriell-Adapter versorgt es auch wie ein normales USB-Kabel mit Strom. Die Programmierung funktioniert genauso wie bei jedem anderen Arduino. Wähle im Menü Tools→Board Arduino Uno oder Arduino Ethernet aus.

Der Ethernet-Shield lässt sich wie viele Shields für den Arduino sehr einfach mit dem Hauptmodul verbinden. Steck ihn einfach drauf. Sowohl das Ethernet-Modul als auch der Shield benutzten die Pins 10, 11, 12 und 13 für die Verbindung zum Ethernet-Controller und die Pins 4, 11, 12 und 13 für die Verbindung zur SD-Karte, du kannst diese Pins also nicht anderweitig als Ein- oder Ausgänge benutzen. Verbinde das Ethernet-Modul über ein Netzwerkkabel mit deinem Router. Abbildung 4-3 zeigt die Verbindungen des Arduino Ethernet-Shields zum Mikrocontroller.

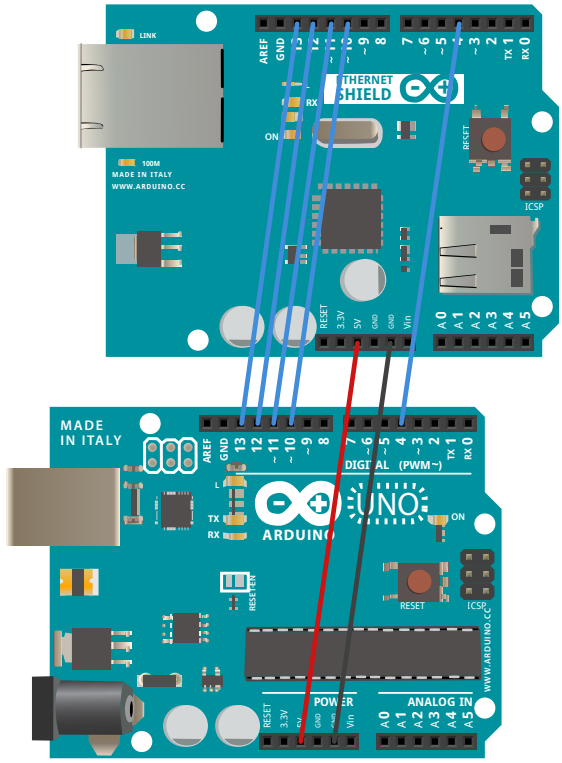


Abbildung 4-3
Die Verbindungen zwischen Ethernet-Shield und dem Arduino-Controller, wenn sie zusammengesteckt sind

Es ist egal, ob du den Arduino Ethernet oder den Ethernet-Shield mit einem Standard-Arduino benutzt, also spreche ich in der Folge nur noch vom „Ethernet-Modul".

Um die Farbe des Umgebungslichts auszulesen, benutzt du drei Photozellen und versiehst jede davon mit einer anderen Farbfolie: rot, grün und blau. Die Folien bekommst du in vielen Beleuchtungsfachgeschäften, Fotoläden und Kunstbedarfsgeschäften, oder du benutzt irgendwelches buntes, durchsichtiges Plastik, das du herumliegen hast. Wir betreiben hier keine genaue Wissenschaft. Abbildung 4-4 zeigt die Verbindung zwischen dem Arduino Ethernet-Modul und den drei Photozellen.

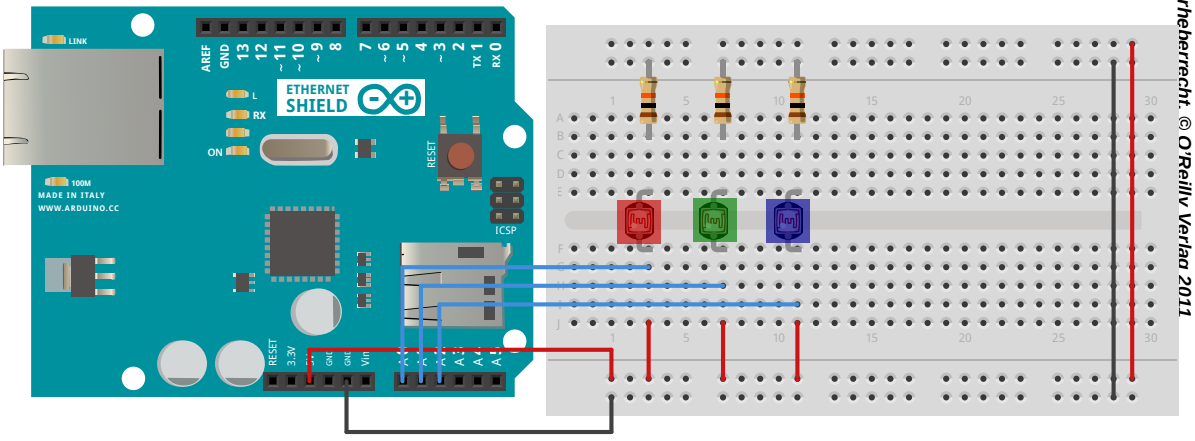
Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628

So funktioniert die Ethernet-Library

Zunächst probierst du einige einfache Programme mit der Ethernet-Library für Arduino aus. Mit dieser Library kannst du das Ethernet-Modul mit Methoden steuern, die denen zur Ausgabe über einen seriellen Port stark ähneln. Es gibt zwei Arten von Software-Objekten, die du erzeugen wirst: Server und Clients. Ein Server wartet auf eingehende Verbindungen von Geräten aus dem Internet und erlaubt ihnen, sich mit dem Ethernet-Modul zu verbinden, genau wie ein normaler Server. Ein Client baut eine Verbindung zu Geräten im Internet auf, stellt Anfragen und liefert Antworten, genau wie ein normaler Client. Man kann sowohl Server als auch Clients auslesen und beschreiben, indem man die Befehle read(), write(), print() und println() benutzt, die wir bereits von der Serial-Library kennen. Es gibt außerdem den Befehl available(), genau wie in der Serial-Library, um festzustellen, ob es vom fraglichen Server oder Client neue Daten gibt. Das Client-Objekt hat auch noch den Befehl connected(), der dir sagt, ob es mit einem Server verbunden ist. Das wird später noch nützlich

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011



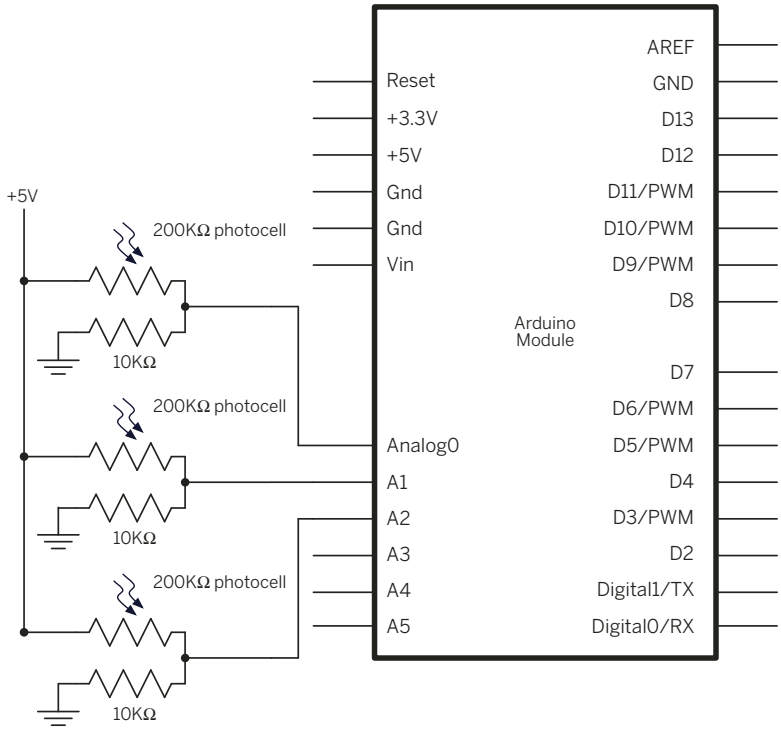


Abbildung 4-4
Die RGB-Server-Schaltung. Die Verbindungen sind beim Arduino Ethernet und bei der Kombination aus Arduino und Ethernet-Shield identisch. Beachte die drei Farbfilter über den Photozellen.

wenn du dich mit einem entfernten Server verbinden willst, um Daten abzufragen. Hier wirst du mit einem einfachen Server-Sketch anfangen.

Bevor du dein Programm schreiben kannst, musst du mit einigen Grundinformationen festlegen, wie sich dein Ethernet-Modul mit dem Internet verbindet. Genau wie in Kapitel 3 brauchst du die MAC-Adresse (Media Access Control) des Moduls. Das ist die Hardware-Adresse deines Ethernet-Controllers. Das Arduino-Ethernet-Modul hat eine aus sechs Bytes bestehende Adresse auf der Rückseite, in hexadezimaler Notation, die du benutzen kannst. Wenn der Aufkleber fehlt, kannst du dir deine eigene MAC-Adresse ausdenken oder eine allgemeine aus den Beispielen weiter unten nehmen. Du musst auch die Adresse des Routers kennen (auch als Gateway-Adresse bezeichnet, weil dein Router das Gateway zum Rest des Internet ist), und außerdem die Adresse, die dein Gerät im Subnetz des Routers benutzen soll.

Die IP-Adresse deines Geräts ist der Adresse deines Routers ähnlich und hat wahrscheinlich dieselben drei Zahlen am Anfang und am Ende dann eine andere Nummer. Wenn die lokale Adresse deines Routers z.B. 192.168.1.1 ist, kannst du eine Adresse wie 192.168.1.20 für dein Ethernet-Modul benutzen, solange kein anderes verbundenes Gerät dieselbe Adresse benutzt.

Wenn ein Router den angeschlossenen Geräten Adressen zuweist, maskiert er einen Teil des Adressraums, so dass diese Geräte nur Adressen im selben Subnetz wie der Router benutzen können. Wenn der Router also z.B. nur Adressen im Bereich von 192.168.1.2 bis 192.168.1.254 zuweist, maskiert er die ersten drei Zahlen (Oktette). Das wird als Netzmaske oder Subnetzmaske bezeichnet. In den Netzwerkeinstellungen an deinem Computer ist diese als ganze Netzwerkadresse ausgeschrieben, etwa so: 255.255.255.0. Du weist sie beim Ethernet-Modul ganz ähnlich zu.

Sobald du deine MAC-Adresse, die Adresse deines Routers, deine IP-Adresse und deine Subnetzmaske kennst, kann es losgehen.

X

Versuch es Zuerst musst du die SPI-Library und die Ethernet-Library einbinden, um das Modul zu steuern. Du musst auch eine Variable initialisieren, die die Server-Instanz beinhaltet. Der Server läuft auf Port 80, genau wie die meisten Webserver.

Du brauchst außerdem vier Variablen für MAC-Adresse, IP-Adresse, Gateway-Adresse und Subnetzmaske. Diese vier speichern wir als Byte-Arrays, in denen jedes Byte in den Adressen in genau einem Byte gespeichert wird.

```
/*
  Webserver
  Kontext: Arduino

 */

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

EthernetServer server(80);

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01 };
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
IPAddress ip(192,168,1,20);
```

» Ändere diese Werte auf die deines Geräts und Netzwerks.

» In der setup()-Methode startest du das Ethernet-Modul und den Server. Öffne außerdem die serielle Verbindung zur Fehlersuche.

```
void setup()
{
  // Starte die Ethernetverbindung und den Server:
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();
  Serial.begin(9600);
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht © O'Reilly Verlag 2011

In der Hauptschleife wartest du die ganze Zeit darauf, dass eine Verbindung von einem entfernten Client hergestellt wird. Wenn du eine Verbindung erhältst, wartest du darauf, dass der Client eine HTTP-Anfrage stellt, wie in Kapitel 3. Diese Schleife antwortet dem Client nicht, zeigt dir aber dessen Anfrage.

```
void loop()
{
  // warte auf eingehende Clients
  EthernetClient client = server.available();
  if (client) {
      while (client.connected()) {
        if (client.available()) {
          char thisChar = client.read();
          Serial.write(thisChar);
        }
      }
    // Verbindung beenden:
    client.stop();
  }
}
```

Verbinde dein Ethernet-Modul mit deinem Router, starte den Sketch und öffne den Serial Monitor. Öffne dann ein Browserfenster und ruf die Adresse des Arduino auf. In diesem Beispiel würdest du http://192.168.1.20 aufrufen. Im Browser siehst du nichts, aber du kannst die ankommende HTTP-Anfrage im Serial Monitor sehen. Jetzt siehst du, was der Server gesehen hat, als du in Kapitel 3 HTTP-Anfragen gestellt hast. Eine typische Anfrage sieht so aus:

```
GET / HTTP/1.1
Host: 192.168.1.1
Connection: keep-alive
Accept: application/xml,application/xhtml+xml,text/
html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X
10_6_5; de-DE) AppleWebKit/534.10 (KHTML, like Gecko)
Chrome/8.0.552.215 Safari/534.10
Accept-Encoding: gzip,deflate,sdch
Accept-Language: de-DE,de;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Du kannst die meisten HTTP-Parameter ignorieren. Zwei Teile sind allerdings hilfreich: Die erste Zeile, in der du siehst, was der Client abfragt, und das Ende, wo durch eine leere Zeile die Anfrage beendet wird. In diesem Fall zeigt die erste Zeile, dass der Client nach der Haupt-Indexseite im obersten Verzeichnis des Servers fragt, was am / in GET / erkennbar ist. In diesem Projekt hast du nur ein Ergebnis, mit dem du anworten kannst, in Zukunft wirst du aber einen komplexeren Server schreiben, der sicht diesen Teil der Anfrage ansieht und je nach Anfrage unterschiedlich antwortet.

Hier kannst du es dir aber einfacher machen und nur auf das Ende der Abfrage achten, die aus einem Zeilenumbruch (\n oder ASCII 10) besteht, auf den ein Wagenrücklauf (\r oder ASCII 13) und noch ein Zeilenumbruch folgen. Abbildung 4-5 zeigt, was im folgenden Code passiert.

X

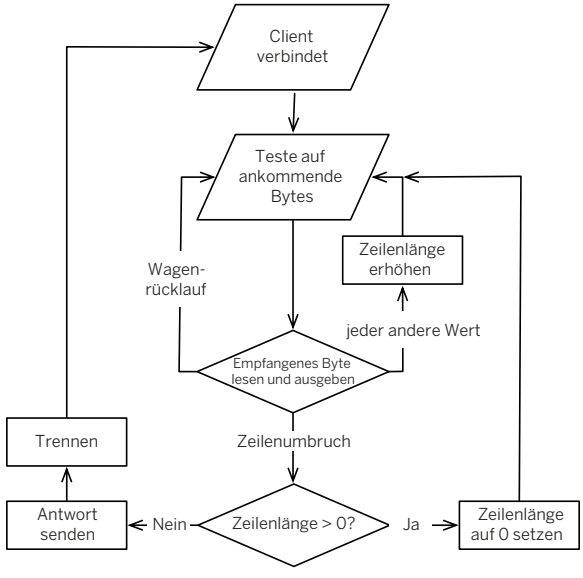


Abbildung 4-5
Prozesslogik eines einfachen Servers

Ändere deine Hauptschleife wie folgt und füge die blauen Zeilen nach Serial.write(thisChar) ein.

```
void loop()
{
  // warte auf eingehende Clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("Habe einen Client");
    String requestLine = "";

    while (client.connected()) {
      if (client.available()) {
        char thisChar = client.read();
        // wenn du einen Zeilenumbruch empfängst und die Anfragezeile
        // leer ist, ist die Anfrage beendet:
        if (thisChar == '\n' && lineLength < 1) {
          // Sende einen Standard-HTTP-Response-Header
          makeResponse(client);
          break;
        }
        //wenn du einen Zeilenumbruch oder Rücklauf empfängst,
        // bist du am Ende einer Zeile:
        if (thisChar == '\n' || thisChar == '\r') {
          lineLength = 0;
        }
        else {
          // vergrößere die Zeilenlänge bei jedem anderen Zeichen:
          lineLength++;
        }
      }
    }
    // gib dem Webbrowser Zeit, die Daten zu empfangen
    delay(1);
    // Verbindung beenden:
    client.stop();
  }
}
```

Bevor du den Code starten kannst, musst du am Ende des Sketchs eine Methode hinzufügen, makeResponse(), die dem Client einen String sendet. Das hier funktioniert erst einmal.

```
void makeResponse(EthernetClient thisClient) {
  thisClient.print("HTTP/1.1 200 OK\n");
  thisClient.print("Content-Type: text/html\n\n");
  thisClient.print("Hallo vom Arduino</head><body>\n");
  thisClient.println("</body></html>\n");
}
```

Wenn du jetzt die Adresse des Arduino in deinen Browser eingibst, bekommst du eine Webseite. Viel ist nicht zu sehen, aber jetzt kann man behaupten, dass dein Browser deinen Arduino nicht von einem beliebigen anderen Webserver unterscheiden kann. Du kannst im HTML in der Methode makeResponse() alles hinzufügen, was du möchtest, auch Links auf Bilder und Inhalte auf anderen Servern. Du kannst dir den Arduino als Portal zu beliebigen anderen Inhalten vorstellen, die du dem User zeigen willst, egal ob es Messdaten von Sensoren oder andere webbasierte Inhalte sind. Du hast die Wahl wie komplex es sein soll. Wenn du erst einmal anfängst, kreativ über die Fähigkeiten der Anweisungen print() und println() nachzudenken, erhältst du eine Vielzahl von Möglichkeiten, dynamisch ein Webinterface für deinen Arduino über den Ethernet-Shield zu erzeugen.

X

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht © O'Reilly Verlag 2011

» Wie wäre es damit, die Werte der Analogeingänge auszugeben?

Füge Folgendes in makeResponse() ein (die neuen Zeilen sind blau).

```
void makeResponse(EthernetClient thisClient) {
  thisClient.print("HTTP/1.1 200 OK\n");
  thisClient.print("Content-Type: text/html\n\n");
  thisClient.print("<html><head>");
  thisClient.print("<title>Hallo vom Arduino</title></head><body>\n");

  // Gib den Wert jedes analogen Pins aus
  for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
    thisClient.print("Analogeingang ");
    thisClient.print(analogChannel);
    thisClient.print(" ist ");
    thisClient.print(analogRead(analogChannel));
    thisClient.print("<br />\n");
  }
  thisClient.println("</body></html>\n");
}
```

» Lade die Seite neu, dann bekommst du die Zustände der Analog-Inputs.

Aber wie wäre es, sie ständig auszulesen? Du kannst die Methoden benutzen, die wir auch für die Katzen-Cam in Kapitel 3 eingesetzt haben. Ändere die Zeile in makeResponse(), die den HTML-Head ausgibt. Siehe den Code rechts.

```
  thisClient.print("Content-Type: text/html\n\n");
  thisClient.print(
    "<html><head><meta http-equiv=\"refresh\" content=\"3\">");
```

» In der Antwort kannst du alle möglichen Dinge tun. Jetzt sollten wir mal die drei an den Analogeingängen angeschlossenen Photozellen einsetzen. Diese Version der makeResponse()-Methode gibt eine Seite aus, deren Hintergrundfarbe sich mit den Werten der drei Photozellen ändert.

Dein Lichtfarben-Server ist fertig, jetzt kannst nach bunten Orten suchen, um ihn aufzustellen.

```
  thisClient.print("Content-Type: text/html\n\n");
  thisClient.print(
    "<html><head><meta http-equiv=\"refresh\" content=\"3\">");
  thisClient.print("<title>Hallo vom Arduino</title></head>");
  // Anfang des body-Tags für Hintergrundfarbe:
  thisClient.print("<body bgcolor=#");
  // Auslesen der drei Analogsensoren:
  int red = analogRead(A0)/4;
  int green = analogRead(A1)/4;
  int blue = analogRead(A2)/4;
  // Ausgabe als hexadezimaler String:
  thisClient.print(red, HEX);
  thisClient.print(green, HEX);
  thisClient.print(blue, HEX);
  // Tag schließen:
  thisClient.print(">");
  // Ausgabe der Farbe im Body der HTML-Seite:
  thisClient.print("Die Lichtfarbe am Arduino ist #");
  thisClient.print(red, HEX);
  thisClient.print(green, HEX);
  thisClient.println(blue, HEX);
  // Seite schließen:
  thisClient.println("</body></html>\n");
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Ein Gerät mit privater IP im Internet sichtbar machen

Bis jetzt haben alle Projekte in diesem Buch, die etwas mit dem Internet zu tun haben, entweder nur im lokalen Subnetz funktioniert oder nur Daten abgesendet und auf eine Antwort gewartet. Dies ist das erste Projekt, bei dem dein Gerät im Internet auch sichtbar sein muss. Du kannst es zwar aufrufen, wenn du im selben lokalen Netzwerk bist, aber wenn es an deinem Heimrouter hängt und nur eine private IP-Adresse hat, ist es für niemanden außerhalb deiner Wohnung sichtbar. Um dieses Problem zu beheben, musst du einen Port auf deinem Router so einrichten, dass er ankommende Nachrichten und Verbindungsanfragen an dein Ethernet-Shield weiterleitet.

Öffne dazu das Administrations-Interface deines Routers und suche nach einer Einstellung für „port forwarding", „Port-Weiterleitung" oder so ähnlich. Das Interface ist je nach Hersteller und Modell deines Routers unterschiedlich, aber es sollte einen solchen Punkt in den Einstellungen geben. Es ist am einfachsten, wenn der weitergeleitete Port am Router derselbe ist wie der offene Port des Ethernet-Moduls. Stelle es also so ein, dass Port 80 an deinem Router an Port 80 deines Ethernet-Shields verbunden wird, wenn dein Router es erlaubt. Sobald du das getan hast, werden alle eingehenden Anfragen an die öffentliche IP-Adresse deines Routers auf diesem Port an die private IP-Adresse des Ethernet-Moduls weitergeleitet, und zwar an den Port, den du eingestellt hast. Es kann sein, dass der Router einige Ports für besondere Anwendungen reserviert hat. Wenn du keine Weiterleitung von Port 80 anlegen kannst, kann es daran liegen, dass der Router diesen für sein eigenes Interface benutzt. Daher musst du vielleicht eine höhere Zahl verwenden, z.B. 8080.

Internetbrowser richten standardmäßig ihre Anfragen an Port 80, aber du kannst eine Anfrage an einen beliebigen Port stellen, indem du die Portnummer ans Ende der Serveradresse anhängst:

```
http://www.myserver.com:8080/
```

Die neue öffentliche Adresse deines Ethernet-Moduls richtet sich dann auch nach diesem System. Wenn die öffentliche Adresse deines Routers z.B. 203.48.192.56 ist, kannst du deinen Arduino-Server unter http://203.48.192.56:8080 erreichen.

Abbildungen 4-6 und 4-7 zeigen die Einstellungen an zwei WLAN-Routern, dem Apple AirPort Express und einem Linksys-Modell. Am Linksys-Router kannst du das Port-Forwarding im Reiter Advanced finden.

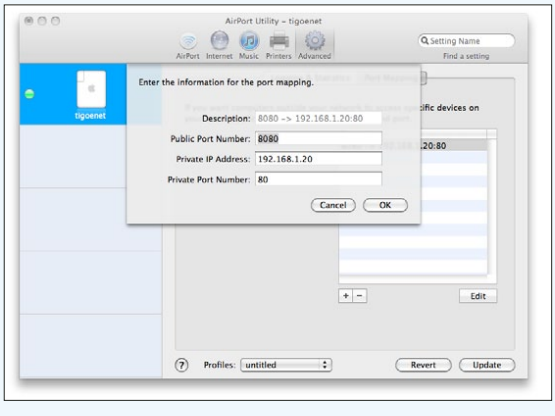


Abbildung 4-6
Reiter „Port mapping" an einem Apple AirPort Express. Das Port Mapping findet sich im Reiter Advanced.

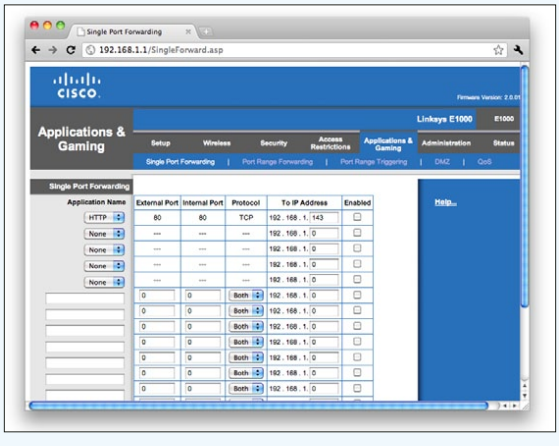


Abbildung 4-7
Port-Weiterleitung an einem WLAN-Router von Linksys

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628

Eine Embedded-Netzwerk-Client-Anwendung

Nachdem du deinen ersten Server gebaut hast, wird es jetzt Zeit für einen Client. Dieses Projekt ist ein eingebetteter Web-Scraper (scrape heißt „kratzen“). Er nimmt Daten von einer existierenden Website, die dann eine physische Ausgabe beeinflussen. Vom Konzept her ähnelt er daher Geräten von Ambient Devices, Nabaztag und anderen, aber du baust ihn selbst.

Projekt 7

Internet-Luftqualitätsmesser

In diesem Projekt wirst du eine Internet-Messstation für Luftqualität bauen. Du brauchst ein analoges Einbau-Messgerät, so wie in einem Tachometer oder einer Hi-Fi-Aussteuerungsanzeige. Ich habe meins vom Flohmarkt, aber du könntest auch in einem Elektronik- oder Gebrauchtwarenladen Glück haben. Das in der Einkaufsliste empfohlene Teil ist nicht so schön wie meins, aber es reicht als Platzhalter, bis du eins findest, das dir gefällt.

Abbildung 4-8 zeigt, wie es funktioniert: Der Mikrocontroller baut über den Ethernet-Shield eine Verbindung zu einem PHP-Skript auf. Das PHP-Skript verbindet sich mit einer anderen Webseite, liest eine Zahl von dieser Seite und schickt die Zahl zurück an den Mikrocontroller. Der Mikrocontroller nimmt diese Zahl und stellt daraufhin die Anzeige des Messgeräts ein. Die eingesetzte Website ist AIRNow, www.airnow.gov, die Seite der US-Umweltschutzbehörde, die über die Luftqualität informiert. Dort wird stündlich neu die Luftgüte in vielen US-Städten nach Postleitzahl (ZIP code) angegeben. Am Ende des Projekts hast du ein Messgerät zu Hause oder im Büro, das dir die momentane Luftqualität in deiner Stadt angibt (sofern du in den USA wohnst).

MATERIALIEN

- **Ein Arduino Ethernet oder**
- **Ein Arduino Ethernet-Shield und ein Arduino-Mikrocontroller**
- **Eine Ethernet-Verbindung zum Internet**
- **Ein Steckbrett**
- **Ein Voltmeter**
- **4 LEDs**
- **4 220-Ω-Widerstände**

Ansteuerung des Messgeräts mit dem Mikrocontroller

Zuerst musst du mit dem Mikrocontroller eine veränderbare Spannung erzeugen, um das Messgerät zu steuern. Mikrocontroller können keine Analogspannungen ausgeben, aber sie können eine Reihe von sehr schnellen Ein-/Aus-Impulsen erzeugen, die gefiltert werden können, um eine Durchschnittsspannung zu ergeben. Je höher das Verhältnis von Signal und Pause in jedem Impuls ist, desto höher ist die Durchschnittsspannung. Diese Technik nennt man Pulsbreitenmodulation, englisch pulse-width modulation (PWM). Damit ein PWM-Signal als Analogspannung wirkt, muss die Schaltung, die die Impulse erhält, viel langsamer reagieren als die Taktrate der Impulse. Wenn du z.B. eine LED mit PWM ansteuerst, sieht es so aus, als ob sie gedimmt wäre, weil dein Auge den Wechsel zwischen Ein- und Ausschalten nicht erkennen kann, wenn er schneller als etwa 30-mal pro Sekunde erfolgt. Analoge Voltmeter reagieren sehr langsam auf Spannunsänderungen, also

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8 http://www.oreilly.de/catalog/makingthingstalkger/ Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

lassen sie sich sehr gut mit PWM ansteuern. Indem du den positiven Anschluss des Messgeräts an einen Ausgangspin des Mikrocontrollers und den negativen Anschluss an Masse anschließt und den Ausgangspin per PWM steuerst, kannst du einfach die Position der Nadel einstellen. Abbildung 4-9 zeigt den gesamten Schaltplan für das Projekt.

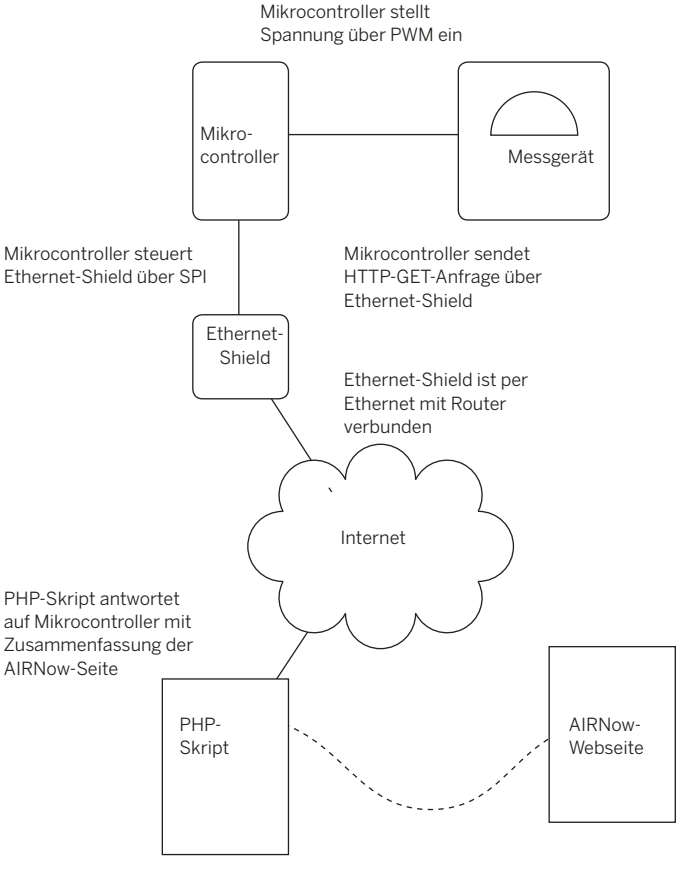
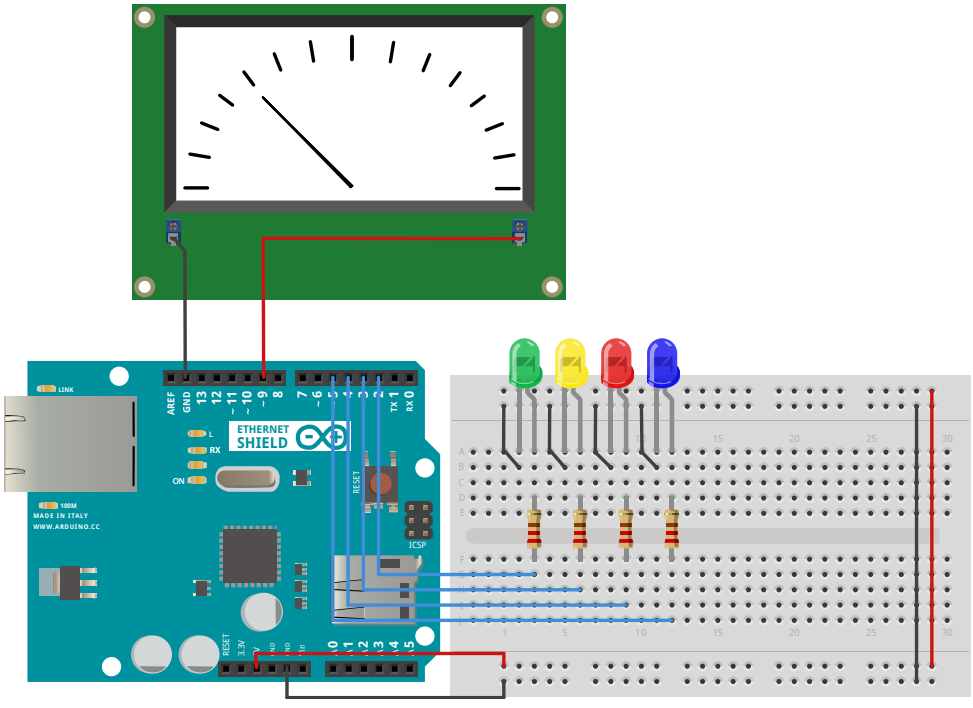


Abbildung 4-8
Die Internet-Messstation für Luftqualität

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011



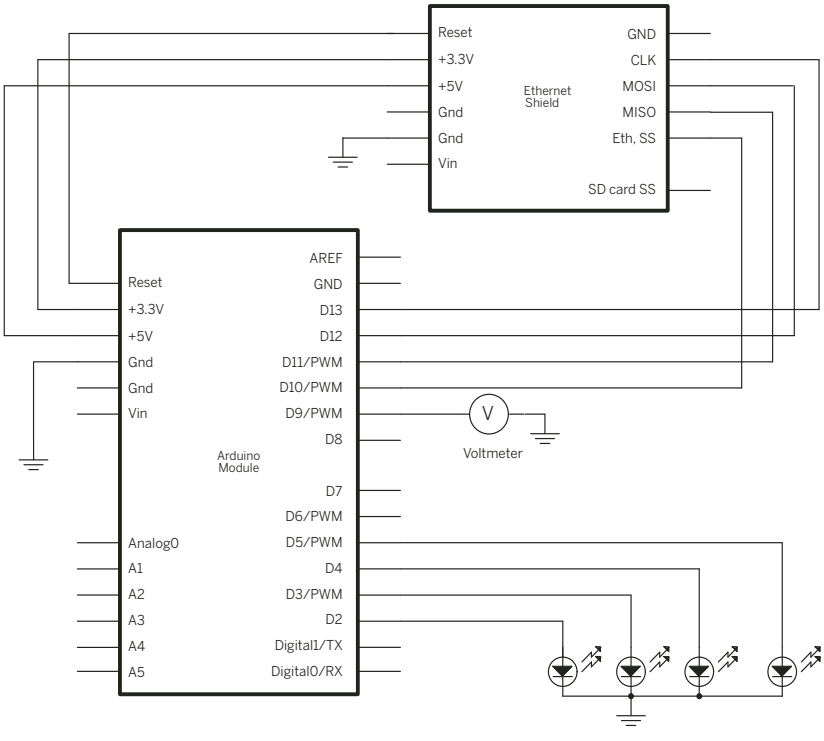


Abbildung 4-9
Die Schaltung für ein vernetztes Messgerät. Der Ethernet-Controller im Schaltplan ist auf dem Shield oder dem Arduino Ethernet-Modul.

Teste es Das Programm rechts testet, ob du das Voltmeter steuern kannst.

Du wirst den Bereich von pwmValue in Abhängigkeit von der Empfindlichkeit des Anzeige einstellen müssen. Die Voltmeter, die ich beim Entwurf dieses Projekts benutzt habe, hatten unterschiedliche Anzeigebereiche. Das Voltmeter aus der Einkaufsliste reagiert auf einen Bereich von 0 bis 5 Volt, das obige Programm bewegt den Zeiger also über die gesamte Skala. Das antike Messgerät reagiert dagegen nur von 0 bis 3 Volt, daher war es nötig, den Bereich von pwmValue auf 0 bis 165 zu begrenzen. Bei einem Wert von 165 erreichte der Zeiger das Maximum. Notier dir den Minimal- und Maximalwert deines Voltmeters. Du brauchst sie später, um die Luftgütemessung auf die Skala abzugleichen.

```
/*
    Voltmeter-Tester
    Steuert ein Voltmeter über analogWrite().
    Kontext: Arduino
*/
const int meterPin = 9;

int pwmValue = 0;  // der Wert für das Voltmeter

void setup() {
  Serial.begin(9600);
}

void loop() {
  // beweg das Voltmeter vom niedrigsten zum höchsten Wert:
  for (pwmValue = 0; pwmValue < 255; pwmValue ++) {
    analogWrite(meterPin, pwmValue);
    Serial.println(pwmValue);
    delay(10);
  }
  delay(1000);
  // Nullstellung und Pause:
  analogWrite(meterPin, 0);
  delay(1000);
}
```

Schreib ein PHP-Skript, um die Webseite auszulesen

Als Nächstes musst du die Daten von der AIRNow-Webseite in eine Form bringen, die der Mikrocontroller lesen kann. Der Mikrocontroller kann kurze Strings seriell einlesen. Es ist auch einfach, diese ASCII-Strings in eine Binärzahl umzuwandeln. Es ist auch möglich, mit einem Mikrocontroller den gesamten Text einer Webseite durchzugehen (zu parsen), aber es ist etwas komplizierter. Aber es ist genau die Aufgabe, für die sich PHP eignet. Das folgende Programm liest die AIRNow-Seite aus, filtert dort den aktuellen Luftgütewert heraus und erzeugt eine einfachere Übersichtsseite, die sich einfach mit dem Mikrocontroller auslesen lässt. Der Ethernet-Controller ist für den Mikrocontroller das Zugangstor zum Internet. Er ermöglicht dir, eine TCP-Verbindung zu deinem Webhoster aufzubauen, wo du dieses PHP-Skript installieren wirst.

Hinweis: Du könntest dieses Skript auch auf einem der Computer in deinem lokalen Netzwerk laufen lassen. Solange der Mikrocontroller mit demselben Netzwerk verbunden ist, kannst du eine Verbindung dorthin aufbauen und die PHP-Seite abrufen. Wie du PHP installierst oder einen Webhoster findest, der PHP unterstützt, erfährst du unter http://www.php.net/manual/de/tutorial.requirements.php.

Abbildung 4-10 zeigt die AIRNow-Seite für New York City (http://airnow.gov/?action=airnow.local_city&zipcode=10003&submit=Go). Die AIRNow-Seite ist so formatiert, dass das Auslesen der Daten nicht schwer fällt. Der AQI-Wert, der die Luftgüte angibt, ist im Text deutlich zu sehen. Wenn du alle HTML-Tags entfernst, steht er auf einer eigenen Zeile, immer nach der Zeile Current Conditions (aktuelle Lage).

Hinweis: Das größte Problem beim Einsatz solcher Anwendungen, die die Daten einer existierenden Website abgreifen, ist die Wahrscheinlichkeit, dass die Designer der Website das Format ihrer Seite ändern. Wenn das passiert, wird deine Anwendung nicht mehr funktionieren und du musst deinen Code umschreiben. Genau das ist tatsächlich zwischen der ersten und zweiten Auflage dieses Buchs passiert. Für solche Fälle ist es sinnvoll, dass man das Auslesen der Webseite über das PHP-Skript durchführt. Es ist praktischer, den PHP-Code umzuschreiben, als den Mikrocontroller umzuprogrammieren, wenn er erst einmal eingebaut ist.

X

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628991628

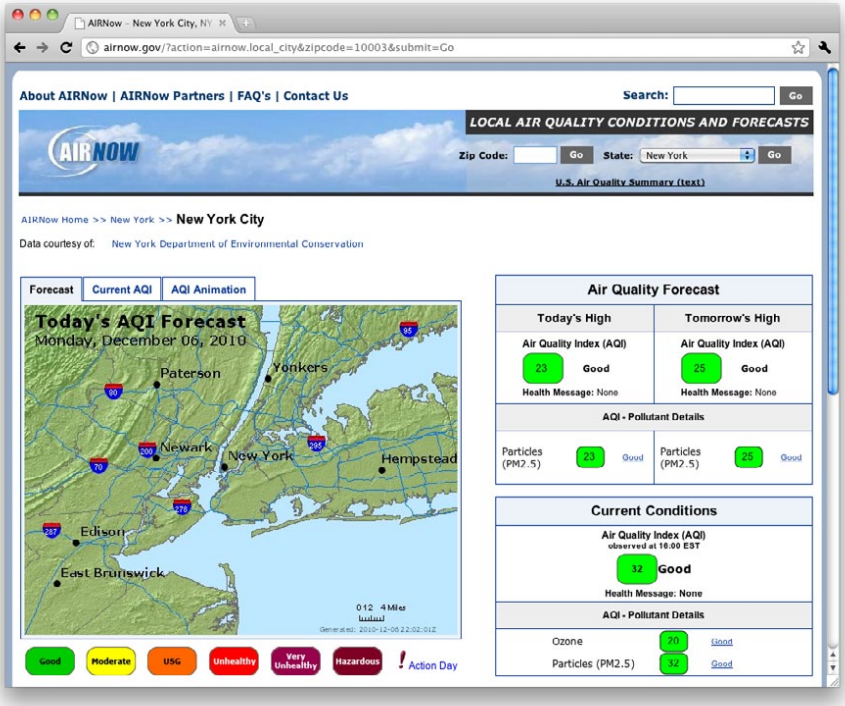


Abbildung 4-10
Die Seite von AIRNow eignet sich gut zum Auslesen. Das PHP-Programm in diesem Projekt ignoriert den Ozonwert.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8 http://www.oreilly.de/catalog/makingthingstalkger/ Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Anfordern

Dieses PHP-Skript öffnet die AIRNow-Webseite und gibt sie Zeile für Zeile aus. Der Befehl fgetss() liest eine Textzeile aus und entfernt alle HTML-Tags.

Wenn du diese Datei auf deinem Webserver abspeicherst und in einem Browser öffnest, solltest du den Text der AIRNow-Seite ohne HTML-Auszeichnungen oder Bilder erhalten. Im Browserfenster ist er nicht gut zu lesen, aber wenn du dir den Quelltext anschaust (benutze die Option „Quelltext anzeigen" in deinem Browser), sieht es etwas besser aus. Wenn du herunterscrollst, findest du Zeilen wie diese:

```
Current Conditions
Air Quality Index (AQI)
observed at 17:00 EST
45
```

Das sind die einzigen Zeilen, die dich interessieren.

```
<?php
/*
    AIRNow-Webseiten-Scraper
    Kontext: PHP
*/
    // Variablen definieren:
    // URL der Seite mit den Luftwerten für New York City:
    $url =
      'http://airnow.gov/index.cfm?action=airnow.showlocal&cityid=164';

    // Öffne die Datei mit dieser URL zum Lesen:
    $filePath = fopen ($url, "r");

    // solange das Dateiende noch nicht erreicht ist:
    while (!feof($filePath))
    {
        // Lies eine Zeile nach der anderen, entferne alle HTML- und
        // PHP-Tags aus der Zeile:
        $line = fgetss($filePath, 4096);
        echo $line;
    }
    // Fertig, schließe die Datei mit der URL:
    fclose($filePath);
?>
```

Auslesen Um die Daten aus diesen Zeilen zu extrahieren, brauchst du noch weitere Variablen. Füge diesen Code vor dem fopen()-Befehl ein.

```
// ob in der aktuellen Zeile ein
// Wert ausgelesen werden muss:
$checkForValue = false;

// Wert der Luftqualität:
$airQuality = -1;
```

» Ersetz den Befehl echo $line; im Programm durch den Codeblock rechts.

Dieser Block benutzt den preg_match()-Befehl, um nach einer Textfolge zu suchen, die einem vorgegebenen Muster entspricht. In diesem Fall sucht er nach dem Muster Current Conditions. Wenn du diese Zeile siehst, weißt du, dass in der nächsten Zeile die gesuchte Nummer steht. Wenn das PHP-Skript diese Zeile findet, setzt es die Variable $checkForValue auf true (wahr).

```
// Enthält die aktuelle Zeile den Substring "Current Conditions"
// dann ist die nächste Zeile mit einer Zahl die Luftqualität:
if (preg_match('/Current Conditions/', $line)) {
   $checkForValue = true;
}
```

» Füge jetzt den folgenden Codeblock hinter dem gerade eingefügten ein. Dieser Code testet, ob $checkForValue wahr ist und diese Zeile eine Integerzahl und sonst nichts enthält. Wenn es so ist, liest das Programm die folgende Zeile und wandelt sie von einem String in einen Integerwert um. Es bekommt nur dann einen gültigen Integerwert, wenn es die Zeile mit dem Luftqualitätsmesswert erreicht.

```
if ($checkForValue == true && (int)$line > 0){
   $airQuality = (int)$line;
   $checkForValue = false;
}
```

» Häng schließlich die folgenden Zeilen am Ende des Skripts nach der while-Schleife an. Hier werden die Luftgütemessung ausgegeben und die Verbindung zur entfernten Site geschlossen.

```
echo "Luftqualität:". $airQuality;
// Fertig, schließ die Datei mit der URL:
fclose($filePath);
```

Das Ergebnis in deinem Webbrowser sollte so aussehen:

```
Air Quality: 43
```

Jetzt hast du einen kurzen Textstring, den dein Mikrocontroller lesen kann. Auch wenn das Skript kein Ergebnis erhalten hat, siehst du am Wert -1, wenn es keine Verbindung gab.

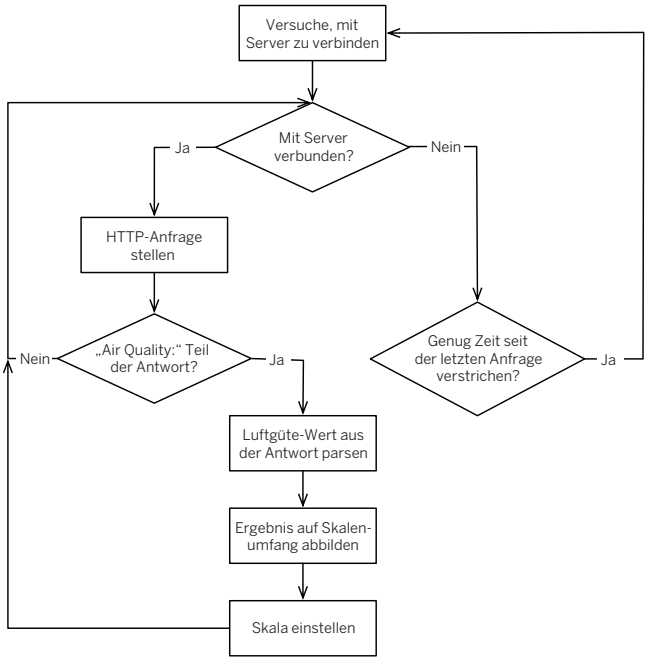
Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Das PHP-Skript mit dem Mikrocontroller auslesen

Als Nächstes stellen wir mit dem Ethernet-Modul über das Internet eine Verbindung zum PHP-Skript her. Dieses Mal benutzt du den Shield als Client, nicht als Server. Plan die Nachrichtensequenz, bevor du zu programmieren beginnst. Die Benutzung des Ethernet-Moduls als Netzwerk-Client ist der Benutzung von Processing als Netzwerk-Client sehr ähnlich. In beiden Fällen musst du die korrekte Abfolge von Nachrichten kennen, die du absendest, und das Format der Antworten. Du musst auch ein Programm schreiben, das den Austausch der Nachrichten erledigt. Egal, ob du das Programm in Processing, in Arduino oder in einer anderen Sprache auf einem anderen Mikrocontroller schreibst, die Schritte sind gleich:

1. Öffne eine Verbindung zu einem Webserver.
2. Sende eine HTTP-GET-Anfrage.
3. Warte auf eine Antwort.
4. Verarbeite die Antwort.
5. Warte eine angemessene Zeit lang und tu das Ganze von vorn.



Abbildun 4-11
Ein Flussdiagramm des Arduino-Programms, um eine HTTP-GET-Abfrage zu stellen und zu verarbeiten

Abbildung 4-11 stellt ein Flussdiagramm der Vorgänge im Mikrocontroller-Programm dar. Die wichtigsten Entscheidungspunkte (if-Anweisungen in deinem Code) sind durch Rauten gekennzeichnet. Die Methoden sind durch Rechtecke gekennzeichnet. Wenn du das ganze Programm in so einem Diagramm aufzeichnest, hilft es dir dabei, nachzuvollziehen, was genau an jeder Stelle passiert. Es hilft auch dabei, einfach zu erkennen, welche Methoden davon abhängen, dass eine bestimmte Bedingung wahr oder falsch ist.

Die Schaltung für dieses Projekt beinhaltet auch LEDs, die den Zustand des Programms anzeigen. LEDs an den Ein-/Ausgabepins zeigen den Status an. Es gibt eine LED, die anzeigt, dass eine Verbindung besteht, eine weitere, die anzeigt, dass die Verbindung nicht besteht, eine dritte, die einen gültigen Auslesevorgang anzeigt und eine vierte, die leuchtet, wenn der Mikrocontroller das Programm zurücksetzt.

Dieses Programm ruft das PHP-Skript alle zwei Minuten ab. Wenn es einen neuen Wert für die Luftgüte gibt, liest es diesen aus und stellt das Messgerät ein. Wenn es keine Verbindung herstellen kann, startet es nach zwei Minuten den nächsten Versuch. Weil es ein Client und kein Server ist, gibt es kein Webinterface zum Projekt, nur das Voltmeter.

TextFinder-Library

Du brauchst für diesen Sketch die TextFinder-Library für Arduino von Michael Margolis. Lad sie von www.arduino.cc/playground/Code/TextFinder herunter, entpack die Datei und kopier das **TextFinder**-Verzeichnis in den Ordner

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

libraries in deinem Arduino-Sketchverzeichnis (standardmäßig in **Dokumente/Arduino/libraries/** unter OS X, **Eigene Dokumente\Arduino\libraries** unter Windows 7 und **~/Documents/Arduino/libraries/** unter Ubuntu Linux. Wenn das Verzeichnis **libraries** nicht existiert, leg es an und kopier TextFinder hinein. Nachdem du Arduino neu gestartet hast, sollte die TextFinder-Libary im Menü Sketch→Import Library auftauchen. Mit TextFinder kannst du einen Teilstring in einem empfangenen Bytestrom finden. Wie du sehen wirst, ist die Library sowohl für Ethernet- als auch für serielle Anwendungen von Nutzen.

Es besteht die Möglichkeit, dass TextFinder in einer zukünftigen Arduino-Version mitgeliefert wird. Dann musst du es nicht extra installieren. Im Bereich Reference von www.arduino.cc gibt es immer die letzten Updates.

Verbinde es Das Programm beginnt wie das vorherige Projekt mit dem Einbinden der SPI- und Ethernet-Libraries. Definiere danach mit Konstanten die Ausgangspins, die Minimal- und Maximalwerte des Voltmeters, die du zuvor festgestellt hast, und die Zeit zwischen HTTP-Anfragen.

```
/*
  AirNow Web-Scraper
 Kontext: Arduino
 */
#include <SPI.h>
#include <Ethernet.h>
#include <TextFinder.h>

const int connectedLED = 2;          // zeigt eine TCP-Verbindung an
const int successLED = 3;            // zeigt erfolgreiches Auslesen an
const int resetLED = 4;              // zeigt Arduino-Reset an
const int disconnectedLED = 5;       // zeigt Verbindung zum Server an
const int meterPin = 9;              // Steuerung des Voltmeters
const int meterMin = 0;              // Minimalwert für Voltmeter
const int meterMax = 200;            // Maximalwert für Voltmeter
const int AQIMax = 200;              // Maximalwert für Luftqualität
const int requestInterval = 10000;   // Pause zwischen zwei Serveranfragen
```

Herausfinden der IP-Adresse eines Hosts

Dieses Projet nutzt numerische IP-Adressen anstelle von Namen. Wenn du die IP-Adresse eines Servers herausfinden willst, benutz den in Kapitel 3 erwähnten `ping`**-Befehl. Wenn du beispielsweise eine Befehlseingabe öffnest und** `ping -c 1 www.oreilly-net.com` **eingibst (tipp unter Windows** `-n` **anstelle von** `-c`**), erhältst du die folgende Antwort, in der die gesuchte IP-Adresse steht:**

```
PING www.oreillynet.com (208.201.239.37): 56 data bytes
64 bytes from 208.201.239.37: icmp_seq=0 ttl=45 time=97.832 ms
```

Wenn du `ping` **nicht benutzen kannst (es wird von einigen Providern blockiert, weil es bösartigen Zwecken dienen kann), kannst du auch** `nslookup` **nehmen. Beispielsweise ergibt** `nslookup google.com` **Folgendes:**

```
Server:  8.8.8.8
Address: 8.8.8.8#53
```

▸▸ nsloopup gibt auch den Domain Name Server aus, über den es die Adresse aufgelöst hat.

```
Non-authoritative answer:
Name:        google.com
Address:     173.194.33.104
```

Alle aufgelisteten Adressen führen zum dazugehörigen Namen, du kannst also irgendeine davon nehmen.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

» Initialisiere als Nächstes einige Arrays mit der Netzwerkkonfiguration des Moduls.

```
byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01 };
IPAddress ip(192,168,1,20);
IPAddress server(208,201,239,101 );
```

» Ändere die Werte, so dass sie deinem Gerät und Server entsprechen.

» Die letzten globalen Variablen, die du noch brauchst, sind für die Client-Klasse und die Transaktion mit dem Server.

```
// Initialisierung der Ethernet-Client-Library
EthernetClient client;

boolean requested;           // wurde eine Anfrage gestellt
long lastAttemptTime = 0;    // letzte Verbindung zum Server
int airQuality = 0;          // Luftgütewert
boolean meterIsSet = false;  // ist das Voltmeter gestellt
```

» setup() startet die serielle und die Ethernet-Verbindung, definiert alle LED-Pins als Ausgänge, lässt die Reset-LED blinken und unternimmt den ersten Versuch, den Server zu kontaktieren.

```
void setup() {
  // Ethernet-Verbindung starten:
  Ethernet.begin(mac, ip);
  // Serial Library starten:
  Serial.begin(9600);
  // Alle Status-LED-Pins setzen:
  pinMode(connectedLED, OUTPUT);
  pinMode(successLED, OUTPUT);
  pinMode(resetLED, OUTPUT);
  pinMode(disconnectedLED, OUTPUT);
  pinMode(meterPin, OUTPUT);

  // eine Sekunde Pause zum Initialisieren des Ethernet-Shields :
  delay(1000);
  // Reset-LED blinken lassen:
  blink(resetLED, 3);
  // Verbindungsversuch:
  connectToServer();
}
```

» Die blink()-Methode, die im Setup aufgerufen wird, lässt die Reset-LED blinken, damit du weißt, dass die Hauptschleife des Mikrocontrollers jetzt beginnt.

```
void blink(int thisPin, int howManyTimes) {
  // Reset-LED blinken lassen:
  for (int blinks=0; blinks< howManyTimes; blinks++) {
    digitalWrite(thisPin, HIGH);
    delay(200);
    digitalWrite(thisPin, LOW);
    delay(200);
  }
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

loop() enthält die gesamte Logik, wie sie in Abbildung 4-11 dargestellt ist. Wenn der Client mit einem Server verbunden ist, stellt er eine HTTP-GET-Anfrage. Sobald er die Anfrage gestellt hat, benutzt er TextFinder, um die Antwort nach dem String mit der Luftgüte zu durchsuchen, und stellt dann die Skala ein. Wenn der Client nicht mit dem Server verbunden ist, wartet er zwei weitere Minuten, bis das Anfrage-Intervall verstrichen ist, und versucht dann erneut, eine Verbindung herzustellen.

```
void loop()
{
  // bei bestehender Verbindung alle Bytes in den
  // Input-String speichern:
  if (client.connected()) {
    if (!requested) {
      requested = makeRequest();
    }
    else {
      // Mit einer Instanz von TextFinder die Antwort durchsuchen:
      TextFinder response(client);
      // Suche, ob die Antwort vom Server den Luftgütewert enthält:
      if(response.find("Air Quality:"))  {
        // restlichen Teil in Integer umwandeln:
       airQuality = response.getValue();
        // Zeiger auf einen Wert stellen:
        meterIsSet = setMeter(airQuality);
      }
    }
  }
  else if (millis() - lastAttemptTime > requestInterval) {
    // Wenn du nicht verbunden bist und seit dem letzten Versuch
    // zwei Minuten vergangen sind, verbinde erneut:
    client.stop();
    connectToServer();
  }

  // Status-LEDs setzen:
  setLeds();
}
```

Sowohl setup() als auch loop() versuchen, mit der Methode connectToServer() eine Verbindung mit dem Server herzustellen. Wenn die Methode eine Verbindung herstellt, setzt sie die Variable requested zurück, damit der Haupt-Loop weiß, dass er eine Anfrage stellen kann.

```
void connectToServer() {
  // Status des Voltmeters löschen:
  meterIsSet = false;

  // Verbindungsversuch, eine Millisekunde warten:
  Serial.println("Verbinde...");
  if (client.connect(server, 80)) {
    requested = false;
  }
  // Zeit des Verbindungsversuchs speichern:
  lastAttemptTime = millis();
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Sobald der Client verbunden ist, sendet er eine HTTP-GET-Anfrage. Hier ist die Methode makeRequest(). Sie gibt einen true-Wert zurück, um die requesting-Variable in der main-Schleife zu setzen, damit der Client während der Verbindung nicht zweimal anfragt.

```
boolean makeRequest() {
  // HTTP-GET-Anfrage stellen und den Pfad zum
  // PHP-Skript auf dem Server einfügen:
  client.println("GET /~myaccount/scraper.php HTTP/1.1\n");
  // Servername einfügen:
  client.print("HOST:example.com\n\n");
  // Status des Programms aktualisieren:
  client.println();
  return true;
}
```

» Ändere diese Werte auf den Namen und Pfad deines Servers.

Der Server antwortet auf makeRequest() mit

```
LufHTTP/1.1 200 OK
Date: Fri, 14 Nov 2010 21:31:37 GMT
Server: Apache/2.0.52 (Red Hat)
Content-Length: 10
Connection: close
Content-Type: text/html; charset=UTF-8

 Air Quality: 65
```

Sobald der Client den String mit der Luftgüte gefunden und die darauf folgenden Bytes in eine Zahl umgewandelt hat, ruft er setMeter() auf, um das Ergebnis auf dem Voltmeter darzustellen. Du solltest noch den Wert AQIMax auf einen typischen Maximalwert der gewünschten Gegend ändern.

```
boolean setMeter(int thisLevel) {
  Serial.println("Zeiger einstellen...");
  boolean result = false;
  // rechne das Ergebnis auf den Bereich der Skala um:
  int meterSetting = map(thisLevel, 0, AQIMax, meterMin, meterMax);
  // Zeiger einstellen:
  analogWrite(meterPin, meterSetting);
  if (meterSetting > 0) {
    result = true;
  }
  return result;
}
```

In der Hauptschleife musst du als letztes die Anzeige-LEDs schalten, damit du weißt, wo du im Programm bist. Du kannst den Status von client.connected() und die Variable meterIsSet benutzen, um die beiden LEDs zu setzen.

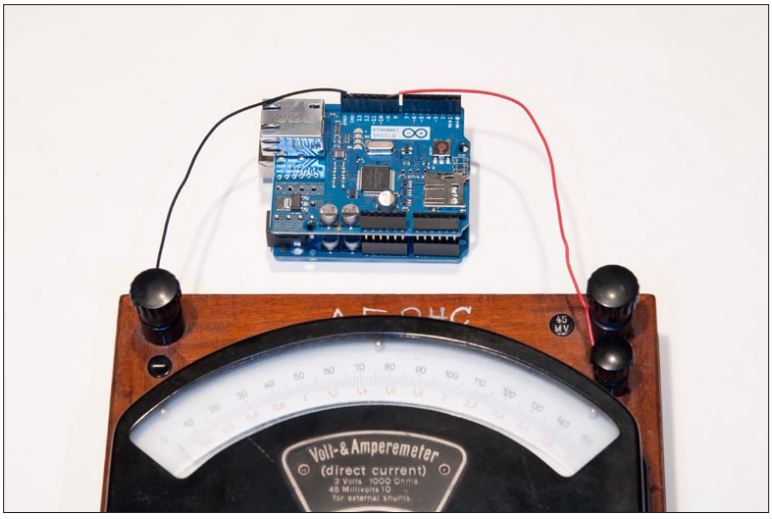
```
void setLeds() {
  // die LEDs für Verbindung und keine Verbindung können
  // einfach den connected()-Status übernehmen:
  digitalWrite(connectedLED, client.connected());
  digitalWrite(disconnectedLED, !client.connected());
  // die Success-LED hängt vom erfolgreichen Auslesen ab:
  digitalWrite(successLED, meterIsSet);
}
```

Das ist das gesamte Programm. Sobald du dieses Programm auf dem Mikrocontroller zum Laufen gebracht hast, wird das Modul alle zehn Minuten die HTTP-GET-Anfrage stellen und das Messgerät entsprechend einstellen.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Das fertige Projekt

Abbildung 4-12
Die fertige Internet-Luftqualitätsanzeige



Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

DNS und DHCP

In Kapitel 3 hast du erfahren, dass die meisten über Ethernet angeschlossenen Geräte über das Dynamic Host Control Protocol (DHCP) eine IP-Adresse von ihrem Router zugewiesen bekommen und dass die über IP verbundenen Geräte die Namen anderer Clients und Server über das Domain Name System (DNS) erfahren. Mit der Arduino-Ethernet-Library kannst du auch DHCP und DNS benutzen. Das geht ganz einfach: Du packst nur die MAC-Adresse in die Ethernet.begin()-Methode:

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
Ethernet.begin(mac);
```

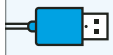
Wenn der Server eine IP-Adresse zuweist, ist der Rückgabewert wahr. Wenn nicht, ist er falsch. Die zugewiesene Adresse wird in Ethernet.localIP() gespeichert. Es gibt noch weitere Variablen für die Benutzung von DHCP, unter anderem Ethernet.subnetMask(), Ethernet.gatewayIP() und Ethernet.dnsServerIP().

Die Benutzung von DNS ist genauso leicht. Anstelle einer Serveradresse als Zahlenfolge übergibst du dem client.connect()-Befehl einfach einen Text:

```
client.connect("www.example.com", 80);
```

Der Nachteil von DHCP und DNS ist, dass dadurch eine Menge RAM belegt wird, daher ist es möglicherweise von Vorteil, die entsprechenden numerischen Adressen zu benutzen, sofern du sie kennst.

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628



DNS und DHCP (Fortsetzung)

Das folgende Beispiel zeigt, wie man eine Adresse über DHCP anfragt; dann wird die erhaltene Adresse ausgegeben:

```
/* DHCP
 Kontext: Arduino
 */
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {
  0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01};
IPAddress ip(192,168,1,20);             // diese Adresse wird benutzt, wenn DHCP fehlschlägt

void setup() {
  // Serial Library starten:
  Serial.begin(9600);
  // Ethernet-Verbindung starten:
  Serial.println("IP-Adresse wird über DHCP angefragt...");
  if (!Ethernet.begin(mac)) {
    // wenn DHCP fehlschlägt, benutze eigene Adresse:
    Ethernet.begin(mac, ip);
  }
  // Ausgabe der IP-Adresse:
  Serial.print("IP-Adresse erhalten. Sie lautet ");
  Serial.println(Ethernet.localIP());
}

void loop() {
}
```

begin() **so zu benutzen, ist in jedem Ethernet-Sketch nützlich, weil damit DHCP benutzt wird, wenn es verfügbar ist, und andernfalls eine manuell gesetzte IP. Wenn du weißt, dass du die Adresse manuell setzen wirst, und weniger Speicher verbrauchen willst, kommentiere einfach die Zeile mit** if() **und die dazugehörige Klammer aus und lass den Befehl dazwischen stehen:**

```
//if (!Ethernet.begin(mac)) {
   // wenn DHCP fehlschlägt, benutze eigene Adresse:
   Ethernet.begin(mac, ip);
// }
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tools zum Programmieren und zur Fehlersuche bei Embedded-Modulen

Beim Herstellen der Verbindungen im letzten Abschnitt hattest du vielleicht einige Probleme. Vermutlich war die größe Schwierigkeit bei der Fehlersuche, dass weder der Ethernet-Shield noch der Mikrocontroller irgendeinen Hinweis darauf geliefert haben, dass ein Problem bestand. Wenn du mit selbst gebauten Embedded-Modulen arbeitest, ist das eher die Norm. Dieser Abschnitt behandelt einige Dinge, die du immer überprüfen solltest, und einige Hilfsmittel, um Probleme zu lösen. Diese Prinzipien treffen zu, egal ob du die Ethernet-Module oder irgendein anderes Netzwerk- oder Kommunikationsmodul benutzt. Du wirst diese Methoden immer wieder benutzen, im weiteren Verlauf des Buchs und darüber hinaus.

Die drei häufigsten Fehler

Überprüfe die Stromversorgung

Überprüfe immer, ob du Stromversorgung und Masse richtig angeschlossen hast. Das ist meistens kein Problem, wenn du ein schönes Steckmodul wie das Ethernet-Shield hast, das sich nur auf eine Weise aufstecken lässt. Es kann dir aber sogar bei solchen Modulen passieren, dass du nicht merkst, wenn ein Pin verbogen ist. Wenn du Glück hast, hat dein Modul Status-LEDs, die leuchten, wenn es korrekt funktioniert. Unabhängig davon solltest du die Spannung zwischen den Polen mit einem Voltmeter überprüfen, um sicherzugehen, dass du es richtig angeschlossen hast.

Überprüfe die Verbindungen

Wenn du ein Modul manuell mit einem Mikrocontroller verbindest, passiert es sehr oft, dass du beim ersten Versuch Kabel vertauschst. Du solltest sicherstellen, dass du weißt, was jeder einzelne Pin tut. Vergewissere dich, dass der Sendepin des einen Geräts mit dem Empfangspin des anderen verbunden ist, und umgekehrt. Wenn es eine synchron-serielle Verbindung ist, dann achte darauf, dass die Clock-Verbindung besteht.

Überprüfe die Konfiguration

Wenn du dir sicher bist, was die Hardware-Verbindungen angeht, überprüfe die Konfiguration des Gerätes. Hast du die IP-Adresse richtig eingegeben? Ist die Router-Adresse richtig? Ist die Netzmaske richtig?

Diagnosetools und -methoden

Wenn du weißt, dass das Gerät funktioniert, musst du die Abfolge von Nachrichten, aus denen deine Anwendung besteht, programmieren. Je nachdem, was für die Anwendung nötig ist, kann diese Abfolge komplex werden. Daher hilft es, wenn man einige einfache Programme zur Hand hat, um zu überprüfen, ob Vorgänge wie gedacht ablaufen.

Es ist eine gute Idee, dein Modul erst mit einem Beispielprogramm zu testen, sofern es eines gibt. Jeder gute Hersteller von Kommunikationsmodulen liefert dir ein Anfangsbeispiel. Du solltest es benutzen und immer zur Hand haben. Wenn irgendetwas falsch läuft, kannst du damit wieder ausprobieren, ob die Grundfunktionen noch laufen.

Sichtbare Debug-Methoden

Durch das Schreiben von Code erzeugt man wahrnehmbare Ereignisse. Das vergisst man leicht, wenn man wie in den Beispielen hier am Austausch von Informationen arbeitet. Es ist also hilfreich, physische Aktionen hinzuzufügen, die du auslösen kannst. Dadurch kannst du sehen, dass dein Code funktioniert. Im letzten Projekt hast du gesehen, wie LEDs angingen, wenn der Client die Verbindung öffnet oder schließt, wenn er eine erfolgreiche Anfrage gestellt hat und wenn er zurückgesetzt wurde. Zu den einfachsten und verlässlichsten Dingen, die du mit einem Mikrocontroller machen kannst, gehört es, eine LED zu schalten. Das ist noch einfacher, als seriell eine Debug-Nachricht zu schicken. Du solltest großzügig LEDs einsetzen, um die Funktion jedes Abschnitt deines Codes sicherzustellen. Später kannst du sie ja immer noch weglassen.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Serielle Debug-Methoden

Der Ethernet-Shield und der Arduino Ethernet machen dir auf verschiedene Weisen das Leben leicht, nicht nur, weil sie so kompakt sind. Beispielsweise benutzen sie synchon-serielle Übertragung, um den Ethernet-Controller anzusteuern. Das heißt, dass du immer noch den seriellen Port frei hast, um Zustandsmeldungen zu senden, wie du im vorherigen Beispiel gesehen hast. Im Folgenden findest du einige Tipps, um Netzwerkcode effektiv über serielle Nachrichten zu debuggen, also um Fehler zu bereinigen.

Dabei solltest du nicht vergesen, dass serielle Übertragungen Zeit kosten. Wenn du serielle Anweisungen in deinem Code hast, verlangsamen sie ihn. Der Mikrocontroller muss jedes empfangene Bit speichern, was deinem Programm die Arbeitszeit wegnimmt. Auch wenn die Zeit für jedes Bit sehr klein ist, addiert sich das auf. Nach ein paar hundert Bytes fällt es dir vielleicht schon auf. Achte also darauf, die Anweisungen für serielle Übertragungen herauszunehmen, wenn du sie nicht mehr brauchst.

Fehler suchen

Eine Methode, das Debuggen zu organisieren, ist mithilfe einer Variablen, die das Verhalten deines Programms verändert, wie hier zu sehen. Wenn jede Debug-Anweisung von der Bedingung if (DEBUG) abhängt, kannst du alle einfach auf einmal abschalten, in dem du DEBUG auf false setzt.

```
const boolean DEBUG = true;

void setup() {
      Serial.begin(9600);
}
void loop() {
      if (DEBUG) Serial.println("Dies ist eine Debug-Ausgabe.");
}
```

Ansagen machen

Im Projekt mit dem Luftgüte-Client gab es viele verschiedene Methoden im Code. Man kann nicht so einfach feststellen, ob wirklich jede Methode aufgerufen wird. Bei der Fehlersuche solltest du es dir also angewöhnen, seriell anzukündigen, wenn eine Methode aufgerufen wird. Setz dafür eine Serial.print-Anweisung an den Anfang jeder Methode, wie rechts im Code zu sehen.

```
void connectToServer() {
      if (DEBUG) Serial.print("starte connectToServer()...");
      // der Rest der Methode folgt hier
}

void makeRequest() {
      if (DEBUG) Serial.print("starte makeRequest()...");
      // der Rest der Methode folgt hier
}

boolean setMeter(int thisLevel) {
      if (DEBUG) Serial.print("starte setMeter()...");
      // der Rest der Methode folgt hier
}
```

Bedingungen prüfen

In einer Kommunikationsanwendung wie dieser ist es normal, dass es verschiedene Bedingungen gibt, die ein bestimmtes Ergebnis beeinflussen. Wenn du arbeitest, machst du schnell auch Fehler, die diese verschachtelten if-Anweisungen beeinflussen. Du solltest immer überprüfen, ob sie tatsächlich wahr werden, wenn du es erwartest. Gib eine Meldung aus, so wie es hier gezeigt ist.

```
if (client.connected()) {
  if (DEBUG) Serial.print("verbunden");
  if (!requested) {
    if (DEBUG) Serial.print("verbunden, nicht angefragt.");
    requested = makeRequest();
  }
  else {
     if (DEBUG) Serial.print("nicht verbunden");
  }
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Aufteilen Programmierer lieben Effizienz, daher kombinieren sie oft mehrere Befehle in einer Codezeile. Manchmal steckt das Problem in der Kombination. Diese Zeile versucht beispielsweise, über client.connect() eine Verbindung herzustellen, und überprüft das Ergebnis dann in einer bedingten Anweisung. Ich hatte damit jede Menge Probleme, bis ich den Verbindungsversuch und die Überprüfung der Verbindung voneinander getrennt habe.

```
if (client.connect()) {
  // über Terminal berichten, wenn die Verbindung steht:
  Serial.println("verbunden");
}
else {
  // wenn keine Verbindung möglich war:
  Serial.println("Verbindung fehlgeschlagen");
}
```

» Hier ist der weniger effiziente Code, der das Problem gelöst hat. Die Verbindung direkt nach dem Verbindungsaufbau zu checken, war offenbar zu früh. Die Verzögerung hat das ganze Programm stabilisiert.

```
client.connect();   // connect
delay(1);           // eine Millisekunde warten
if (client.connected()) {
  // über Terminal berichten, wenn die Verbindung steht:
  Serial.println("verbunden");
}
else {
  // wenn keine Verbindung möglich war:
Serial.println("Verbindung fehlgeschlagen");
}
```

Einfach zusehen Manchmal hilft es auch, von deinem komplizierten Programm einen Schritt Abstand zu nehmen und einfach zu beobachten, was du von der Gegenseite empfängst. Eine Methode, die ich bei der Entwicklung dieses Programms immer wieder eingesetzt habe, war einfach auszugeben, was das Ethernet-Modul empfangen hat. Dadurch bin ich schon mehrmals auf die Lösung gestoßen. Manchmal lag es daran, dass ich in meinem Code logische Probleme hatte. In anderen Fällen hatte ich ein Zeichen übersehen, das der Server als Teil des Protokolls sendet. Vertrautheit kann einen dafür blind machen, was man sehen müsste. Ruf dir also ins Gedächtnis, was du tatsächlich empfängst, im Gegensatz zu dem, was du zu empfangen glaubst.

```
// bei bestehender Verbindung alle Bytes in den
// Input-String speichern:
if (client.connected()) {
  if (client.available()) {
    char inChar = client.read();
    Serial.write(inChar);
  }
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628989783868991628

Schreib einen Test-Client

Am einfachsten arbeitet man sich durch die Schritte des Programms, indem man die Abfolge der Ereignisse nachzeichnet. Komplexere Entwicklungsumgebungen erlauben dir, Programme Zeile für Zeile durchzuschalten. Diese Möglichkeit gibt es bei Arduino nicht, aber man kann andere Umgebungen damit verbinden. Das folgende Programm reicht einfach alles, was am seriellen Port ankommt, an den Ethernet-Port weiter und umgekehrt. Damit wird dein Arduino zu einem Seriell-Ethernet-Wandler. Mit diesem Code kannst du eine Verbindung zu einem seriellen Terminal oder Processing (oder irgendeiner anderen Entwicklungsumgebung, die seriell kommunizieren kann) aufbauen, um die Ethernet-Verbindung zu testen.

Teste es Dieses Programm ist so nützlich, weil du den Nachrichtenaustausch manuell testen oder aber ein Programm in einer Entwicklungsumgebung schreiben kannst. Sobald du die Abfolge richtig hinbekommen hast, kannst du sie in Code für das Arduino-Modul übersetzen.

```
/*
  Seriell-zu-Ethernet
  Kontext: Arduino
 */

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

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01 };
IPAddress ip(192,168,1,20);
IPAddress server(208,201,239,101 );

// Initialisierung der Ethernet-Client-Library
// mit der IP-Adresse und dem Port des Servers
// mit dem du verbinden willst (Port 80 ist Standard für HTTP):
EthernetClient client;

void setup() {
  // Serial Library starten:
  Serial.begin(9600);
  // Ethernet-Verbindung starten:
  if (!Ethernet.begin(mac)) {
    Serial.println("DHCP fehlgeschlagen, konfiguriere manuell.");
    // manuelle Konfiguration mit deiner IP-Adesse:
    Ethernet.begin(mac, ip);
  }
  // eine Sekunde Pause zum Initialisieren des Ethernet-Shields:
  delay(1000);
  Serial.println("Startklar.");
}

void loop() {
  if (client.connected()) {
    // Wenn die Verbindung steht, Bytes vom Client nach seriell leiten:
    if (client.available()) {
      char netChar = client.read();
      Serial.write(netChar);
    }
    // Bytes von seriell zum Client leiten:
    if (Serial.available()) {
      char serialChar = Serial.read();
```

» Ändere die Werte, so dass sie deinem Gerät und Server entsprechen.

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Fortsetzung von der vorherigen Seite

```
      client.write(serialChar);
    }
  }
  else {
    // falls du verbunden warst, Client stoppen:
    client.stop();
    // wenn du nicht verbunden bist und ein
    // C seriell erhältst: Verbindungsversuch:
    if (Serial.available()) {
      char serialChar = Serial.read();
      if (serialChar == 'C') {
        connectToServer();
      }
    }
  }
}

void connectToServer() {
  // Verbindungsversuch, eine Millisekunde warten:
  Serial.println("Verbinde...");
  if (client.connect(server, 80)) {
    // über Terminal berichten, wenn die Verbindung steht:
    Serial.println("verbunden");

  }
  else {
    // wenn keine Verbindung möglich war:
    Serial.println("Verbindung fehlgeschlagen");
    client.stop();
  }
}
```

Kommuniziere Es folgt ein einfacher Processing-Sketch, der seriell mit dem vorherigen Arduino-Sketch kommuniziert, um eine HTTP-Anfrage zu stellen. Beim ersten Tastendruck sendet er C, um eine Verbindung aufzubauen. Der zweite Tastendruck sendet die HTTP-Anfrage.

```
/*
 Seriell-zu-Ethernet: HTTP-Anfragetester
 Kontext: Processing

 */
// Serial Library einbinden
import processing.serial.*;

Serial myPort;       // Serial-Objekt
int step = 0;        // aktueller Schritt im Ablauf
char linefeed = 10;  // ASCII-Zeilenvorschub

void setup()
{
  // Liste der seriellen Ports abrufen:
  println(Serial.list());
```

»

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628991628

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-9
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Fortsetzung von der vorherigen Seite

```
  // öffne den richtigen seriellen Port auf deinem Computer:
  myPort = new Serial(this, Serial.list()[0], 9600);
  // Konfiguriere das Serial-Object, den Text zu puffern,
  // bis ein Zeilenumbruch empfangen wird:
  myPort.bufferUntil(linefeed);
}

void draw()
{
  // keine Aktion im draw-Loop
}

void serialEvent(Serial myPort) {
  // jeden seriell empfangenen String im Fenster ausgeben
  print(myPort.readString());
}

void keyReleased() {
  // wenn eine Taste gedrückt wird, nächster Schritt:
  switch (step) {
  case 0:
    // Öffne eine Verbindung zum angegebenen Server:
    myPort.write("C");
    // den Schritt eins weiterzählen, damit der nächste Tastendruck
    // weiterschaltet:
    step++;
    break;
  case 1:
    // HTTP-GET-Anfrage senden
    myPort.write("GET /~myaccount/index.html HTTP/1.1\n");
    myPort.write("HOST:myserver.com\n\n");
    step++;
    break;
  }
}
```

▸▸ Ändere die Werte, so dass sie deinem Server entsprechen.

Von diesem Sketch ausgehend kannst du die gesamte Logik des HTTP-Austauschs in Processing entwickeln und dann deinen Sketch in einen Arduino-Sketch umwandeln.

Dieser Ansatz erlaubt dir, die logische Abfolge einer Anwendung in einer Umgebung herauszuarbeiten, die dir vielleicht vertrauter ist. Damit vermeidet man die Komplikationen der Mikrocontroller-Entwicklung und kann sich auf die Abfolge der Nachrichten und die Aktionen, die diese auslösen, konzentrieren.

Wenn du so arbeitest, solltest du dich nicht zu sehr in den technischen Details deines Programms verlieren. Du musst dich ja trotzdem später mit den Details des Mikrocontroller-Programms auseinandersetzen. Es hilft aber oft auch, die Unterschiede zwischen zwei Programmiersprachen und ihre Gemeinsamkeiten zu erkennen, wenn du an einem Problem arbeitest. Dadurch kannst du dich besser auf die Logik konzentrieren, die ihnen zugrunde liegt.

X

Schreibe einen Testserver

Das vorige Programm ermöglichte dir, die Verbindung zu einem entfernten Server aufzubauen und den Nachrichtenaustausch zu testen. Der entfernte Server steht jedoch außerhalb deiner Kontrolle, du kannst also nicht mit Sicherheit sagen, dass er deine Nachrichten überhaupt empfangen hat. Wenn du noch nie eine Verbindung hergestellt hast, kannst du gar nicht wissen, ob sich das Modul überhaupt mit einem Server verbinden kann. Um das zu testen, schreib dein eigenes Serverprogramm, mit dem es sich verbinden kann.

Dies ist ein kurzes Processing-Programm, das du auf deinem PC laufen lassen kannst. Es wartet auf eingehende Verbindungen und gibt alle Nachrichten aus, die über diese Verbindungen gesendet werden. Es antwortet mit einer HTTP-Rückmeldung und einer einfachen Webseite.

Wenn du es benutzt, achte darauf, dass dein Ethernet-Modul und dein PC im selben Netzwerk sind. Starte dann dieses Programm und stell eine Verbindung über einen Browser her, mit der URL http://localhost:80.

Wenn du die Funktion überprüft hast, schreib ein Client-Programm für den Arduino, der sich mit der IP deines PCs auf Port 80 verbindet. Dann kannst du beide Seiten der Kommunikation beobachten und herausfinden, wo das Problem liegt. Wenn die Nachrichten in diesem Programm in der richtigen Reihenfolge ankommen, ändere einfach den Connect-String im Code auf dem Mikrocontroller auf den Webserver, mit dem du dich verbinden willst – dann sollte es gehen. Wenn es nicht klappt, liegt das Problem wahrscheinlich bei deinem Webserver. Kontaktiere deinen Webhoster, um zu erfahren, wie du die Diagnosetools auf den Servern benutzen kannst, insbesondere alle Error-Logs deines Servers.

X

Wenn das bei dir nicht funktioniert, ändere den Port auf 8080, was eine gängige Alternative bei vielen Webservern ist. Wenn du einen Webserver auf deinem PC laufen lässt, musst du ggf. die Portnummer in diesem Programm ändern.

```
/*
  Testserver-Programm
 Kontext: Processing

 Erzeugt einen Server, der auf Clients wartet und
 ausgibt, was diese übermitteln. Es sendet auch dem letzten Client alles,
 was auf der Tastatur eingegeben wird.
 */

// Net-Library einbinden:
import processing.net.*;

int port = 80;        // der Port, auf dem der Server wartet
Server myServer;      // das Server-Objekt
int counter = 0;
void setup()
{
  myServer = new Server(this, port); // Server starten
}

void draw()
{
  // den nächsten Client holen, der etwas schickt:
  Client thisClient = myServer.available();
  // wenn die Nachricht nicht leer ist, gib aus, was gesendet wird:

  if (thisClient != null) {
    // ankommende Bytes vom Client lesen:
    while(thisClient.available() > 0) {
      print(char(thisClient.read()));
    }
    // eine HTTP-Antwort schicken:
    thisClient.write("HTTP/1.1 200 OK\r\n");
    thisClient.write("Content-Type: text/html\r\n\r\n");
    thisClient.write("<html><head><title>Hallo</title></head>");
    thisClient.write("<body>Hallo Client! " + counter);
    thisClient.write("</body></html>\r\n\r\n");
    // Verbindung trennen:
    thisClient.stop();
    counter++;
  }
}
```

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011

Tom Igoe, Making Things Talk, O´Reilly, ISBN 9783868991628

Fazit

Die Projekte, die wir in diesem Kapitel behandelt haben, stehen für ein sehr flexibles und nützliches Konzept von vernetzten Objekten. Ein Objekt ist im Grunde ein Browser oder ein Server. Es fragt Informationen aus dem Netz ab und extrahiert daraus die benötigten Daten, oder es liefert Informationen an einen Client. Diese Konzepte kannst du in vielen verschiedenen Projekten einsetzen.

Der Vorteil dieses Ansatzes ist, dass es nicht viel Arbeit ist, bestehende Webanwendungen umzufunktionieren. Um die relevanten Informationen einer bestehenden Website zusammenzufassen, musst du nicht mehr tun, als den PHP-Webscraper aus diesem Kapitel umzuschreiben. Diese Flexibilität vereinfacht es Mikrocontroller-Enthusiasten, die keine große Erfahrung in Webentwicklung haben, mit Webentwicklern zusammenarbeiten, und umgekehrt. Es erleichtert auch die Benutzung der Arbeit von anderen, wenn du keinen Mitstreiter findest.

Dieser Ansatz hat auch Grenzen. In Kapitel 5 lernst du Methoden kennen, um diese Grenzen mit einem anderen Ansatz zu umgehen. Auch wenn du nicht dieses Konzept einsetzt, vergiss nicht die hier erwähnten Werkzeuge zur Problemlösung. Du kannst dir das Leben leichter machen, indem du einfache Dummy-Programme an einem Ende deiner Übertragung ausprobierst. Damit kannst du nämlich sehen, was passieren sollte, und dann das, was tatsächlich passiert, verändern, damit es stimmt.

X

Dies ist ein Auszug aus dem Buch "Making Things Talk ", ISBN 978-3-86899-162-8
http://www.oreilly.de/catalog/makingthingstalkger/
Dieser Auszug unterliegt dem Urheberrecht. © O'Reilly Verlag 2011