9 Eine Universalfernbedienung konstruieren

Fernbedienungen sind sehr komfortabel, aber nicht ohne Fehler. Manchmal fehlt eine bestimmte Funktion, die Sie gern hätten, z. B. eine Zeitschaltung für den Standby-Modus. Außerdem scheinen sich Fernbedienungen wie Kaninchen zu vermehren. Sehr schnell übersäen sie den ganzen Couchtisch und müssen von Ihnen mit teuren Batterien gefüttert werden, die Sie natürlich nicht im Haus haben, wenn am Sonntagabend das wichtige Fußballspiel gezeigt wird. Universalfernbedienungen können ein wenig Abhilfe schaffen, aber selbst die teuersten Geräte dieser Art sind nicht vollkommen.

Obwohl wir täglich Fernbedienungen verwenden, wissen doch die wenigsten von uns, wie sie funktionieren. In diesem Kapitel sehen wir uns ausführlich an, wie Fernbedienungen aufgebaut sind. Anschließend bauen Sie eine eigene universelle Fernbedienung, die besser ist als alle, die es im Laden zu kaufen gibt, da Sie sie voll und ganz an Ihre Bedürfnisse anpassen können. Sie können sehr leicht alle Ihre Lieblingsfunktionen hinzufügen und sogar solche einbauen, die andere Fernbedienungen nicht haben. Wenn ein kommerzielles Produkt die Geräte eines bestimmten Herstellers nicht erkennt, haben Sie normalerweise Pech gehabt. Ihrer eigenen Fernbedienung können Sie ganz einfach neue Protokolle hinzufügen. Es ist sogar möglich, zur Übertragung nicht nur Infrarot zu verwenden, sondern auch andere Technologien wie Bluetooth oder WLAN.

Als Erstes lernen Sie die Grundlagen von Infrarotsignalen kennen. Dann werden Sie sehr schnell Ihr erstes Projekt mit einem Infrarotsensor erstellen, um die Steuercodes von irgendeiner Fernbedienung abzugreifen, die Sie zur Hand haben. Wenn Sie diese Steuercodes haben, können Sie sie mit einer Infrarot-LED ausstrahlen und mit der Konstruktion Ihrer eigenen Universalfernbedienung beginnen.

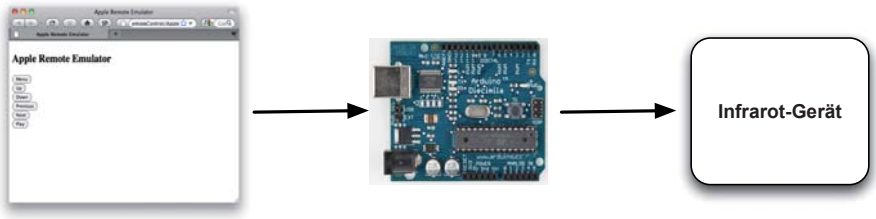

Abb. 9–1 *Architektur des Infrarotproxys*

Anschließend bauen wir die Fernbedienung noch weiter aus. Nachdem wir eine Universalfernbedienung haben, steuern wir Arduino selbst über den seriellen Port oder eine Ethernet-Verbindung. Dadurch können Sie Arduino mit einem Webbrowser kontrollieren. So wird es möglich, Ihren Fernseher oder Ihren DVD-Recorder über das Internet zu bedienen (siehe Abbildung 9–1).

9.1 Was Sie benötigen

- ein Ethernet-Shield für Arduino ①
- ein Breadboard ②
- einen Infrarotempfänger, vorzugsweise PNA 4602 ③
- einen 100-Ohm-Widerstand ④
- eine Infrarot-LED ⑤
- einige Kabel ⑥
- eine oder mehrere Infrarotfernbedienungen von Ihrem Fernseher, DVD-Player oder Ihrem Mac. Um die Beispiele in diesem Kapitel nachzuvollziehen, ist es am besten, wenn Sie einen Mac und eine Apple-Fernbedienung verwenden, aber unbedingt notwendig ist es nicht. Wenn Sie eine andere Fernbedienung einsetzen, müssen Sie den Protokollnamen, die Bitlänge und die Steuercodes in den Beispielen entsprechend anpassen. Setzen Sie bei einer Fernbedienung für einen Sony-Fernseher beispielsweise das Protokoll auf SONY (mehr darüber erfahren Sie im Abschnitt 9.3, *Steuercodes abgreifen*). ⑦ sowie
- ein Arduino-Board wie das Uno, Duemilanove oder Diecimila
- ein USB-Kabel, um Arduino mit Ihrem Computer zu verbinden

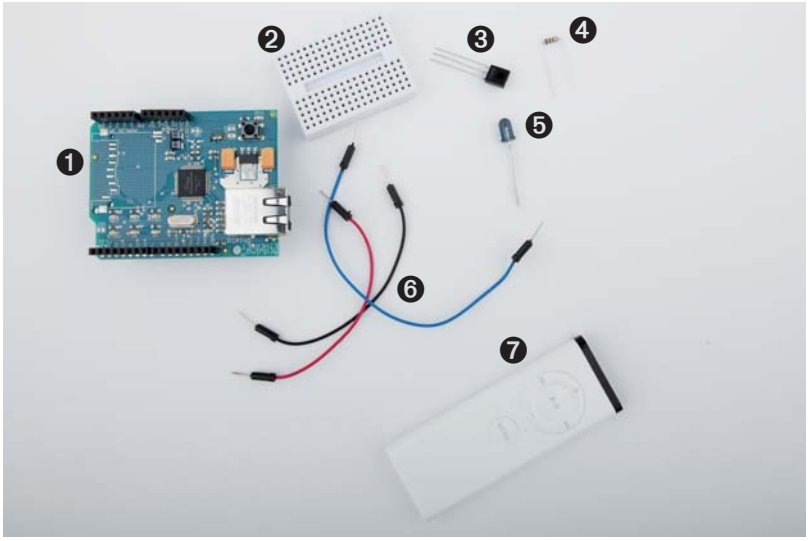

Abb. 9–2 *Die für dieses Kapitel benötigten Bauteile*

9.2 Die Grundlagen von Infrarot-Fernbedienungen

Um ein Gerät wie einen Fernseher drahtlos zu steuern, werden ein Sender und ein Empfänger benötigt. Der Empfänger ist gewöhnlich in das zu steuernde Gerät eingebaut, während der Sender zur Fernbedienung gehört. Es stehen zwar eine Reihe von Technologien wie Bluetooth und WLAN zur Auswahl, doch bei den meisten modernen Fernbedienungen wird nach wie vor Infrarotlicht zur Kommunikation verwendet.

Die Verwendung von Infrarotlicht zur Übertragung von Signalen bietet mehrere Vorteile: Es ist für das menschliche Auge unsichtbar, wirkt also nicht störend. Außerdem können Sie es billig mit Infrarot-LEDs erzeugen, die sich leicht in elektronische Schaltungen einbauen lassen. Für viele Zwecke, z. B. für die Fernsteuerung von Geräten in einem typischen Haushalt, ist es daher eine hervorragende Wahl.

Es weist aber auch einige Nachteile auf. Es wirkt nicht durch Wände und Türen hindurch, und die Entfernung zwischen der Fernbedienung und dem ferngesteuerten Gerät ist stark eingeschränkt. Was aber noch wichtiger ist: Infrarotsignale können von anderen Lichtquellen gestört werden.

Um solche Störungen auf ein Minimum zu reduzieren, werden die Infrarotsignale moduliert. Das heißt, dass die LED mit einer bestimmten Frequenz ein- und ausgeschaltet wird, gewöhnlich zwischen 36 und 40 kHz.

Das ist eines der Probleme, die es ein bisschen schwierig machen, eine solide Infrarot-Fernbedienung zu bauen. Das größte Problem besteht jedoch darin, dass die Gerätehersteller unzählige miteinander unvereinbare Protokolle erfunden haben. Alle nutzen unterschiedliche Frequenzen und interpretieren die Daten auf andere Weise. Für einige ist »Licht an« ein Bit mit dem Wert 1, während es für andere 0 bedeutet. Jeder Hersteller definiert seine eigenen Befehle, die unterschiedliche Längen aufweisen können. Um erfolgreich mit den verschiedenen Protokollen für Fernbedienungen arbeiten zu können, müssen wir einen Weg finden, um all diese Eigenschaften für eine bestimmte Fernbedienung in Erfahrung zu bringen.

Um diese Informationen zu gewinnen, gehen wir ganz pragmatisch vor. In den nächsten beiden Abschnitten lernen Sie, wie Sie die Infrarotsignale einer kommerziellen Fernbedienung lesen und wie Sie sie reproduzieren.

9.3 Steuercodes abgreifen

Fernbedienungen unterschiedlicher Hersteller verwenden nur selten die gleichen Protokolle oder Befehle. Bevor wir eigene Fernsteuercodes senden können, müssen wir daher zunächst in Erfahrung bringen, was wir senden müssen, um ein bestimmtes Ergebnis zu erzielen. Wir müssen so viele Informationen wie möglich über die Fernsteuerung sammeln, die wir nachahmen möchten.

Um die Steuercodes für ein bestimmtes Gerät in Erfahrung zu bringen, haben wir zwei Möglichkeiten: Wir können in einer entsprechenden Datenbank im

Internet wie dem Linux-Projekt *Infrared Remote Control* nachschlagen[1] oder sie mit einem Infrarotempfänger direkt von der Fernbedienung des betreffenden Geräts ablesen. Im Folgenden verwenden wir die zweite Vorgehensweise, da wir dabei sehr viel lernen können.

Infrarotempfänger (siehe Abbildung 9–3) sind sehr kompliziert aufgebaut, aber einfach zu verwenden. Sie beobachten automatisch einen Frequenzbereich für infrarotes Licht (gewöhnlich zwischen 26 und 40 kHz) und melden ihre Wahrnehmungen über einen einzelnen Pin. Wenn Sie einen solchen Empfänger verwenden, haben Sie also nichts mit den komplizierten Einzelheiten der Übertragung zu tun, sondern können sich darauf konzentrieren, die eingehenden Signale zu lesen und zu deuten.

Abb. 9–3 *Infrarotsensor PNA 4602*

In Abbildung 9–4 können Sie sehen, wie Sie einen PNA 4602-Empfänger an Arduino anschließen. Er ist einfach, leicht zu verwenden und läuft auf der Frequenz 38 kHz, so dass er Signale von einer breiten Palette an Geräten auffangen kann. Verbinden Sie seinen Masseanschluss mit einem der GND-Pins von Arduino, den Stromanschluss mit dem 5-V-Pin von Arduino und den Signal-Pin mit dem Digital-Pin 11.

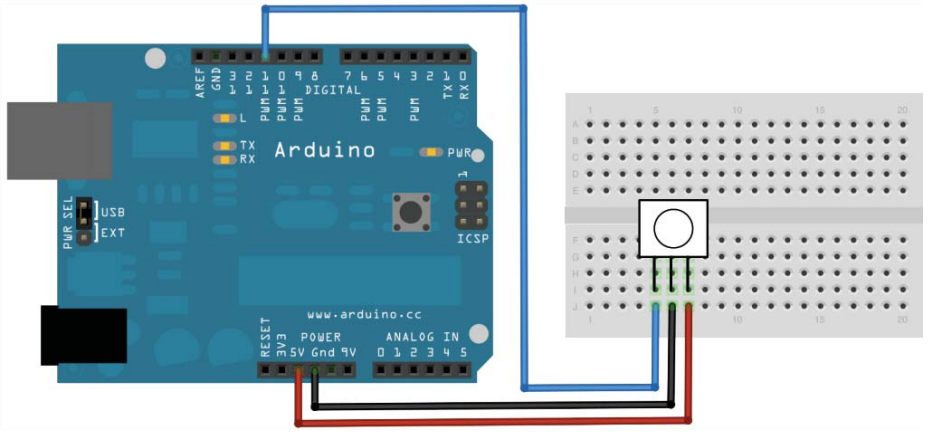

Abb. 9–4 *Der Anschluss eines IR-Empfängers an Arduino ist einfach.*

1 *http://www.lirc.org/*

Sie könnten versucht sein, einen Sketch zu schreiben, der alle auf Pin 11 eingehenden Daten liest und ausgibt, und ich werde Sie nicht daran hindern. Rufen Sie *digitalRead* in der Methode *loop* auf, um die Ergebnisse am seriellen Port auszugeben. Richten Sie die Fernbedienung Ihres Fernsehers auf den Empfänger und beobachten Sie, was passiert.

Wahrscheinlich wird es Ihnen schwerfallen, die angezeigten Daten zu verstehen. Die Dekodierung der eingehenden Daten ist nicht einfach. Auch wenn der Empfänger die Daten bereits verarbeitet hat, müssen sie immer noch anhand von komplizierten Regeln umgeformt und interpretiert werden. Außerdem ist die Arduino-Methode *digitalRead* nicht genau genug, um mit allen Arten von eingehenden Signalen umgehen zu können. Um die bestmöglichen Ergebnisse erzielen zu können, brauchen Sie direkten Zugang zum Microcontroller.

Glücklicherweise müssen wir das nicht selbst machen. Die Bibliothek IRremote[2] schirmt uns von diesen nervtötenden Kleinigkeiten ab. Sie beherrscht die meisten bekannten Infrarot-Protokolle und kann Daten sowohl empfangen als auch senden. Nachdem Sie die ZIP-Datei[3] heruntergeladen und ausgepackt haben, kopieren Sie das Verzeichnis *IRremote* in den Ordner *~/Documents/Arduino/libraries* (auf einem Mac) oder in *Eigene Dokumente\Arduino\Libraries* (Windows). Starten Sie die IDE dann neu.

Mit dem folgenden Sketch können Sie eingehende Infrarotsignale dekodieren, wenn die Bibliothek *IRremote* die zugehörige Kodierung kennt.

RemoteControl/InfraredDumper/InfraredDumper.pde

```
Zeile 1   #include <IRremote.h>
-
-         const unsigned int IR_RECEIVER_PIN = 11;
-         const unsigned int BAUD_RATE = 9600;
5
-         IRrecv ir_receiver(IR_RECEIVER_PIN);
-         decode_results results;
-
-         void setup() {
10          Serial.begin(BAUD_RATE);
-           ir_receiver.enableIRIn();
-         }
-
-         void dump(const decode_results* results) {
15          const int protocol = results->decode_type;
-           Serial.print("Protocol: ");
-           if (protocol == UNKNOWN) {
-             Serial.println("not recognized.");
-           } else {
20            if (protocol == NEC) {
-               Serial.println("NEC");
-             } else if (protocol == SONY) {
-               Serial.println("SONY");
-             } else if (protocol == RC5) {
```

2 *http://www.arcfn.com/2009/08/multi-protocol-infrared-remote-library.html*

3 *http://arcfn.com/files/IRremote.zip*

```
25              Serial.println("RC5");
-             } else if (protocol == RC6) {
-               Serial.println("RC6");
-             }
-             Serial.print("Value: ");
30            Serial.print(results->value, HEX);
-             Serial.print(" (");
-             Serial.print(results->bits, DEC);
-             Serial.println(" bits)");
-           }
35        }
-
-         void loop() {
-           if (ir_receiver.decode(&results)) {
-             dump(&results);
40            ir_receiver.resume();
-           }
-         }
```

Als Erstes definieren wir ein *IRrecv*-Objekt namens *ir_receiver,* das Daten von Pin 11 liest. Außerdem definieren wir ein *decode_result*-Objekt zur Speicherung der Attribute eingehender Infrarotsignale. In *setup* initialisieren wir den seriellen Port. Den Infrarotempfänger initialisieren wir durch den Aufruf von *enableIRIn*.

Anschließend definieren wir die Methode *dump*, die den Inhalt eines *decode_result*-Objekts sauber formatiert und am seriellen Port ausgibt. *decode_result* ist einer der Kerndatentypen der Bibliothek *IRremote*. Er kapselt Daten wie den Protokolltyp, die Länge eines Befehlscodes und den Befehlscode selbst. In Zeile 15 lesen wir den Protokolltyp aus, der zur Kodierung des eingehenden Signals verwendet wurde. Sobald wir ein neues Signal empfangen, geben wir all diese Attribute am seriellen Port aus.

Die Methode *loop* ist einfach. Wir rufen *decode* auf, um zu prüfen, ob ein neues Signal eingegangen ist. Wenn ja, rufen wir *dump* auf, um es am seriellen Port auszugeben. Anschließend rufen wir *resume* auf, um auf das nächste Signal zu warten.

Kompilieren Sie den Sketch und laden Sie ihn zu Arduino hoch. Starten Sie dann den seriellen Monitor und richten Sie eine Fernbedienung auf den Empfänger. Drücken Sie einige der Tasten auf der Fernbedienung und beobachten Sie, was passiert. In Abbildung 9–5 sehen Sie beispielsweise, was geschieht, wenn Sie eine Apple-Fernbedienung auf den Empfänger richten und die Tasten »Menü«, »Nach oben«, »Nach unten«, »Zurück«, »Vor« und »Wiedergabe« drücken. (Wenn Sie den Code 0xffffffff sehen, haben Sie eine der Tasten der Fernbedienung zu lange gedrückt. Dies ist der »Wiederholungscode«, der angibt, dass der letzte Befehl wiederholt werden soll.)

Nachdem Sie die Steuercodes einer Fernbedienung abgegriffen haben, können Sie sie für den Bau Ihrer eigenen Fernbedienung einsetzen. Wie das geht, lernen Sie im nächsten Abschnitt.

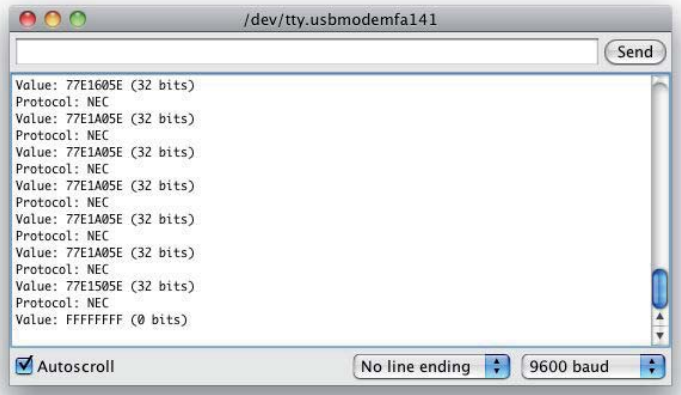

Abb. 9–5 *Die IR-Codes einer Apple-Fernbedienung erfassen*

9.4 Eine eigene Apple-Fernbedienung bauen

Da Sie jetzt das Protokoll und die Befehlscodes kennen, die die Apple-Fernbedienung an einen Mac sendet, können Sie eine eigene Fernbedienung dieser Art bauen. Sie brauchen nur noch eine Infrarot-LED, die den bisher verwendeten LEDs sehr stark ähnelt. Der einzige Unterschied besteht darin, dass sie »unsichtbares« Licht aussendet. In Abbildung 9–6 sehen Sie, wie Sie sie an Pin 3 von Arduino anschließen. (Die Bibliothek, die wir in diesem Abschnitt verwenden, erwartet, dass die Infrarot-LED an diesen Pin angeschlossen ist.) Beachten Sie, dass Sie eine LED nicht ohne Widerstand verwenden können (mehr dazu erfahren Sie im Abschnitt *Stromstärke, Spannung und Widerstand* von Anhang A).

Wir könnten versuchen, die Infrarotsignale selbst zu generieren, aber das wäre sehr mühselig und fehleranfällig. Am besten ist es, die vorhandene Implementierung in der Bibliothek IRremote zu verwenden. Wir nutzen sie, um unsere eigene *AppleRemote*-Klasse zu erstellen, die alle fiesen Einzelheiten des Protokolls kapselt. Die Klasse sieht wie folgt aus:

RemoteControl/AppleRemote/AppleRemote.pde

```
#include <IRremote.h>

class AppleRemote {

  enum {
    CMD_LEN = 32,
    UP      = 0x77E15061,
    DOWN    = 0x77E13061,
    PLAY    = 0x77E1A05E,
    PREV    = 0x77E1905E,
```

```
    NEXT    = 0x77E1605E,
    MENU    = 0x77E1C05E
  };

  IRsend mac;

  void send_command(const long command) {
    mac.sendNEC(command, CMD_LEN);
  }

  public:

  void menu() { send_command(MENU); }
  void play() { send_command(PLAY); }
  void prev() { send_command(PREV); }
  void next() { send_command(NEXT); }
  void up() { send_command(UP); }
  void down() { send_command(DOWN); }
};
```

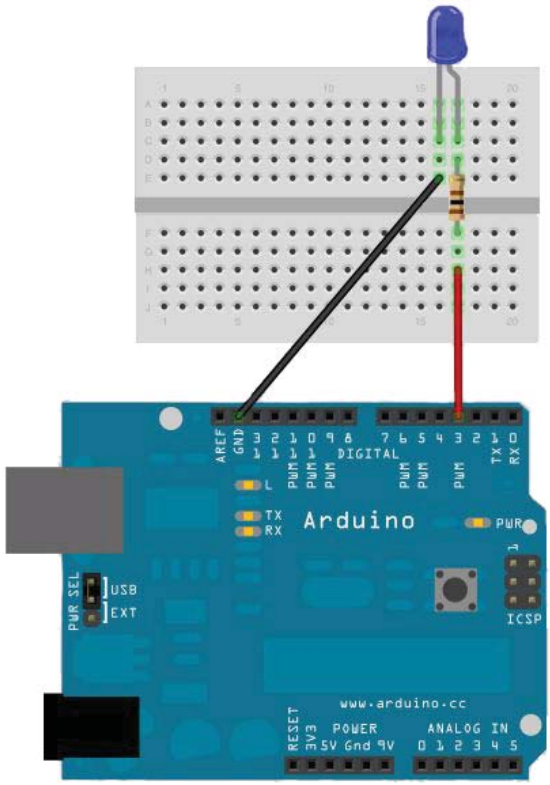

Abb. 9–6 *Eine IR-LED an Arduino anschließen*

Der Code beginnt mit einer Aufzählung, die alle benötigten Konstanten enthält, nämlich die Länge der einzelnen Steuerbefehle und die Steuerbefehle selbst. Dann

definieren wir das *IRsend*-Objekt *mac*, das wir verwenden, um Befehle mithilfe der Methode *send_command* zu senden. *send_command* verwendet die Methode *sendNEC* von *IRsend*, da die Apple-Fernbedienung auf das Protokoll NEC zurückgreift.

Nachdem wir die Grundlage gelegt haben, können wir alle Befehle mit einem einzigen Funktionsaufruf implementieren. Die Programmierung von *menu*, *play* usw. wird dadurch zum Kinderspiel.

Auch die Verwendung der Klasse *AppleRemote* ist einfach. Im folgenden Sketch nutzen wir sie, um einen Mac vom seriellen Arduino-Monitor aus zu steuern.

RemoteControl/AppleRemote/AppleRemote.pde

```
AppleRemote apple_remote;
const unsigned int BAUD_RATE = 9600;

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

void loop() {
  if (Serial.available()) {
    const char command = Serial.read();
    switch(command) {
      case 'm':
        apple_remote.menu();
        break;
      case 'u':
        apple_remote.up();
        break;
      case 'd':
        apple_remote.down();
        break;
      case 'l':
        apple_remote.prev();
        break;
      case 'r':
        apple_remote.next();
        break;
      case 'p':
        apple_remote.play();
        break;
      default:
        break;
    }
  }
}
```

Wir definieren das globale *AppleRemote*-Objekt *apple_remote* und initialisieren den seriellen Port in der Funktion *setup*. In *loop* warten wir auf neue Daten am seriellen Port. Sobald ein neues Byte eintrifft, überprüfen wir, ob es sich um eines der Zeichen m, u, d, l, r oder p handelt. Je nachdem, welches Zeichen wir empfangen, senden wir den Steuercode für »Menü« »Nach oben« (up), »Nach unten« (down), »Zurück« (previous), »Weiter« (next) oder »Wiedergabe« (play).

Kompilieren Sie den Sketch und laden Sie ihn hoch. Jetzt können Sie über einen seriellen Monitor einen Mac steuern, was schon eine coole Sache ist. Für Leute, die nicht gerade ausgebuffte Computerfreaks sind, wirkt die Oberfläche jedoch immer noch ein bisschen unbeholfen. Im nächsten Abschnitt lernen Sie, wie Sie sie benutzerfreundlicher gestalten können.

9.5 Geräte im Browser fernsteuern

Wir haben bereits eine Menge Projekte erstellt, die Sie mit einem seriellen Monitor steuern können. Für Programmierer ist das eine brauchbare und bequeme Oberfläche, aber wenn Sie Ihre Projekte Ihrem Lebenspartner oder Freunden vorstellen möchten, die technisch nicht versiert sind, sollten Sie lieber eine benutzerfreundlichere und buntere Darstellung wählen.

Das Plug-In Seriality[4] macht dies möglich, indem es dem JavaScript-Modul Ihres Webbrowsers Unterstützung für die Kommunikation mit dem seriellen Port hinzufügt.

Zurzeit gibt es dieses Plug-In nur für Firefox, Safari und Chrome auf Mac OS X, aber eine Windows-Portierung befindet sich bereits in der Entwicklung. Seriality ist als Datenträgerimage erhältlich, das Sie herunterladen[5] und wie üblich installieren können.

Nach der Installation von Seriality können Sie mithilfe des folgenden kombinierten HTML- und JavaScript-Codes aus Ihrem Webbrowser einen Simulator für eine Apple-Fernbedienung machen:

RemoteControl/AppleRemoteUI/ui.html

```
Zeile 1   <html>
-           <title>Apple Remote Emulator</title>
-           <head>
-             <script type="text/javascript">
5              var serial;
-
-              function setup() {
-                serial = (document.getElementById("seriality")).Seriality();
-                alert(serial.ports.join("\n"));
10               serial.begin(serial.ports[0], 9600);
-              }
-             </script>
-           </head>
-
15          <body onload="setup();">
-             <object type="application/Seriality"
-                     id="seriality"
-                     width="0"
-                     height="0">
20           </object>
-            <h2>Apple Remote Emulator</h2>
```

4 *http://www.zambetti.com/projects/seriality/*

5 *http://code.google.com/p/seriality/downloads/list*

```
-             <form>
-               <button type="button" onclick="serial.write('m');">
-                 Menu
25             </button>
-               <br/>
-               <button type="button" onclick="serial.write('u');">
-                 Up
-               </button>
30             <br/>
-               <button type="button" onclick="serial.write('d');">
-                 Down
-               </button>
-               <br/>
35             <button type="button" onclick="serial.write('l');">
-                 Previous
-               </button>
-               <br/>
-               <button type="button" onclick="serial.write('n');">
40               Next
-               </button>
-               <br/>
-               <button type="button" onclick="serial.write('p');">
-                 Play
45             </button>
-               <br/>
-             </form>
-           </body>
-         </html>
```

Dies ist eine sehr einfache HTML-Seite, weshalb wir uns im Folgenden auf die JavaScript-Anteile konzentrieren. In den Zeilen 4 bis 12 definieren wir zwei Dinge: nämlich die globale Variable *serial* und die Funktion *setup*. *setup* initialisiert die Variable *serial* und weist ihr ein *Seriality*-Objekt zu. Mit dem <*object*>-Tag betten wir ein *Seriality*-Objekt in die Webseite ein. Dessen ID lautet *seriality*, so dass wir mit *getElementById* darauf zugreifen können.

Sobald wir einen Verweis auf das Objekt haben, rufen wir die JavaScript-Funktion *alert* auf und geben alle gefundenen seriellen Ports aus. Sie müssen den Index in der Liste der seriellen Ports nachschlagen, die mit Arduino verbunden sind, und diesen Index im folgenden Aufruf der Methode *begin* nutzen. Der Einfachheit halber übergeben wir stets das erste gefundene serielle Gerät und eine Baudrate von 9600. Dass das erste serielle Gerät das richtige ist, ist nur geraten. Unter Umständen müssen Sie dies anpassen. Diese Vorgehensweise kennen Sie schon von den Processing-Beispielen.

Im Ereignishandler *onload* des <*body*>-Elements rufen wir *setup* auf. Dann können wir in den *onclick*-Handlern unserer sechs <*button*>-Elemente auf das *Seriality*-Objekt zugreifen.

Laden Sie den Sketch aus dem Abschnitt 9.4, *Eine eigene Apple-Fernbedienung bauen*, zu Arduino hoch und öffnen Sie die HTML-Seite in Ihrem Browser. Nachdem Sie auf die Schaltfläche *OK* des Meldungsfensters geklickt haben, das alle

seriellen Ports anzeigt, sollten Sie eine Webseite wie in Abbildung 9–7 sehen. Klicken Sie auf eine beliebige Schaltfläche, um die zugehörige Aktion durchzuführen. Diese Oberfläche könnte sogar Ihre Großmutter verwenden, oder?

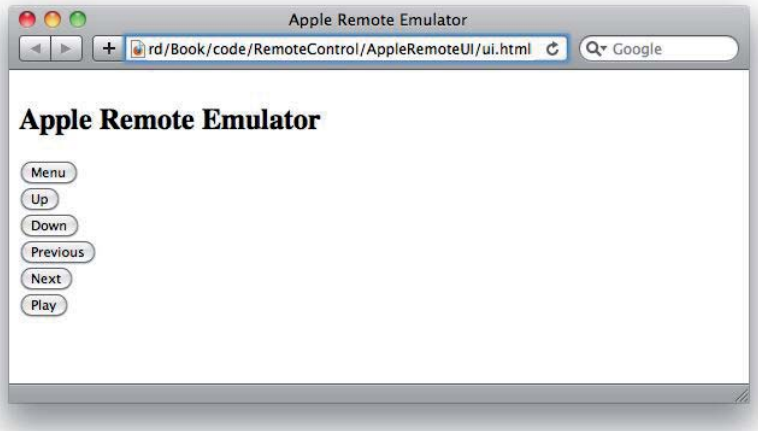

Abb. 9–7 *Der Emulator für die Apple-Fernbedienung im Einsatz*

Beachten Sie, dass es mit Seriality nicht möglich ist, direkt auf die Arduino-Hardware zuzugreifen. Sie können nur auf den seriellen Port zugreifen. Alles, was auf Arduino geschehen soll, muss daher über die serielle Kommunikation zugänglich sein. Das ist aber ohnehin eine übliche Vorgehensweise, so dass Seriality sich als wirklich nützliches Werkzeug erweist, um die Benutzeroberfläche Ihrer Projekte deutlich zu verbessern.

Um Arduino über einen Webbrowser steuern zu können, müssen Sie das Board nach wie vor an den seriellen Port des Computers anschließen. Im nächsten Abschnitt lernen Sie, wie Sie diese Einschränkung überwinden und Arduino ohne serielle Verbindung steuern können.

9.6 Einen Infrarotproxy bauen

Unsere vorhergehenden Konstruktionsversuche für eine Fernsteuerung wiesen alle einen großen Nachteil auf: die Notwendigkeit einer seriellen Verbindung zu einem PC. In diesem Abschnitt lernen Sie, wie Sie die serielle durch eine Ethernet-Verbindung ersetzen, so dass Sie keinen PC mehr benötigen, sondern nur noch einen Internetanschluss. Dabei schließen Sie das Ethernet-Kabel direkt an einem Ethernet-Shield an, der mit Arduino verbunden ist (siehe Abbildung 9–8). Damit ist Arduino in Ihrem Netzwerk verfügbar!

Das heißt nicht, dass Sie unbedingt den Webbrowser Ihres PCs verwenden müssen, um auf Arduino zuzugreifen. Stattdessen können Sie auch den Browser einer PlayStation Portable, eines iPhones oder eines Nintendo DS benutzen. Ja,

Sie können Ihren Fernseher jetzt über Ihre Spielkonsolen oder Ihr Smartphone bedienen! Außerdem können Sie statt des Ethernet- auch einen WiFi-Shield nehmen, so dass Sie Arduino nicht mehr physisch mit Ihrem Netzwerkrouter verbinden müssen.

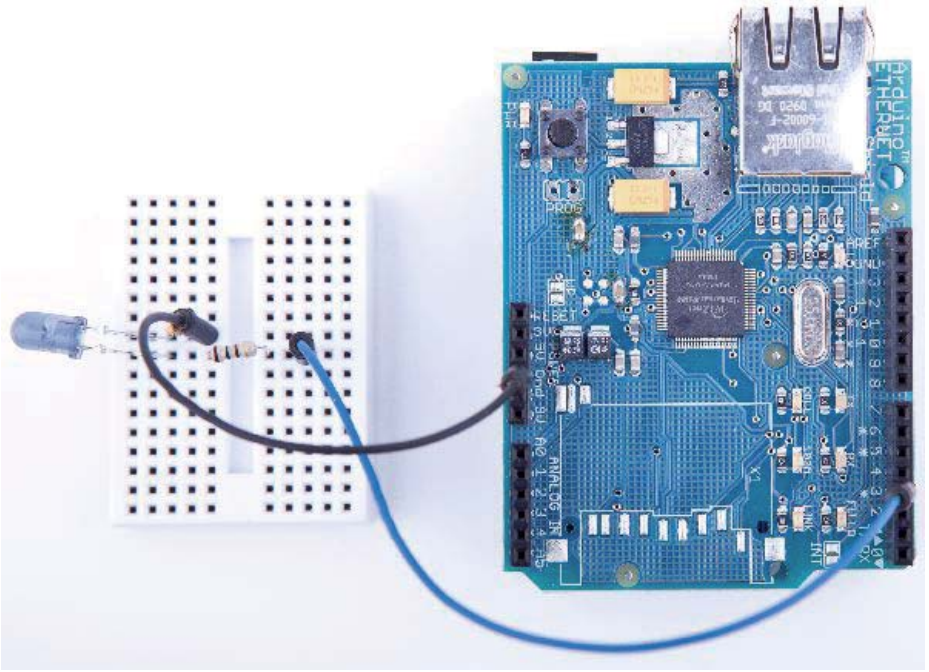

Abb. 9–8 *Eine über Ethernet steuerbare Fernbedienung*

Bevor wir uns genauer mit dem Code beschäftigen, führen wir zunächst eine Planung durch und machen deutlich, was wir erreichen wollen: Wir werden einen Infrarotproxy bauen, also ein Gerät, das Befehle über Ethernet empfängt und in Infrarotsignale umwandelt (siehe Abb. 9–1 auf Seite 165). Um das Gerät einfach in ein Netzwerk einfügen zu können, machen wir es über HTTP zugänglich. Dadurch können wir es über einen normalen Webbrowser steuern.

Wir implementieren nur einen sehr kleinen Teil des HTTP-Standards, da wir nur ein bestimmtes URL-Schema unterstützen. Die möglichen URLs sehen wie folgt aus:

http://arduino-ip/protocol-name/command-length/command-code

Dabei ersetzen wir *arduino-ip* durch die IP-Adresse des Ethernet-Shields. Das Element *protocol-name* kann eines der unterstützten Protokolle sein (NEC, SONY, RC5 oder RC6). *command-length* gibt die Länge der Befehlscodes in Bits an, und *command-code* enthält den eigentlichen Befehlscode in Form einer Dezimalzahl.

Nehmen wir an, wir wollen den Befehl für die Menütaste auf einer Apple-Fernbedienung senden. Unser Arduino soll die IP-Adresse 192.168.2.42 haben. Dann müssen wir im Webbrowser folgenden URL eingeben:

http://192.168.2.42/NEC/32/2011283550

In diesem Fall lautet der Protokollname NEC, die Länge des Befehlscodes beträgt 32 Bits, und der Befehlscode selbst ist 2011283550 (die dezimale Darstellung der Hexadezimalzahl 0x77E1C05E).

Im Kapitel 8, *Netzwerkeln mit Arduino,* haben wir Arduino bereits als Webclient genutzt, aber jetzt machen wir daraus einen Webserver. Der Server wartet wie der bereits gezeigte auf neue HTTP-Anforderungen, analysiert den URL und gibt das entsprechende Infrarotsignal ab.

Die Einzelheiten verbergen wir in der Klasse *InfraredProxy*. Um die Sache so einfach und knapp zu halten wie möglich, verwenden wir sowohl die Ethernet- als auch die IRremote-Bibliothek. Die Klasse *InfraredProxy* ist trotzdem noch eines der anspruchsvollsten Beispiele von Arduino-Code in diesem Buch. Und so sieht sie aus:

RemoteControl/InfraredProxy/InfraredProxy.pde

```
Zeile 1   #include <SPI.h>
-         #include <Ethernet.h>
-         #include <IRremote.h>
-
5         class InfraredProxy {
-           IRsend _infrared_sender;
-
-           void read_line(Client& client, char* buffer,
                           const int buffer_length) {
-             int buffer_pos = 0;
10            while (client.available() && (buffer_pos < buffer_length - 1)) {
-               const char c = client.read();
-               if (c == '\n')
-                 break;
-               if (c != '\r')
15                buffer[buffer_pos++] = c;
-             }
-             buffer[buffer_pos] = '\0';
-           }
-
20          bool send_ir_data(const char* protocol,
                              const int bits, const long value) {
-             bool result = true;
-             if (!strcasecmp(protocol, "NEC"))
-               _infrared_sender.sendNEC(value, bits);
-             else if (!strcasecmp(protocol, "SONY"))
25             _infrared_sender.sendSony(value, bits);
-             else if (!strcasecmp(protocol, "RC5"))
-               _infrared_sender.sendRC5(value, bits);
-             else if (!strcasecmp(protocol, "RC6"))
-               _infrared_sender.sendRC6(value, bits);
```

```
30            else
-               result = false;
-             return result;
-           }
-
35          bool handle_command(char* line) {
-             strsep(&line, " ");
-             char* path = strsep(&line, " ");
-
-             char* args[3];
40            for (char** ap = args; (*ap = strsep(&path, "/")) != NULL;)
-               if (**ap != '\0')
-                 if (++ap >= &args[3])
-                   break;
-             const int bits = atoi(args[1]);
45            const long value = atol(args[2]);
-             return send_ir_data(args[0], bits, value);
-           }
-
-           public:
50
-           void receive_from_server(Server server) {
-             const int MAX_LINE = 256;
-             char line[MAX_LINE];
-             Client client = server.available();
55            if (client) {
-               while (client.connected()) {
-                 if (client.available()) {
-                   read_line(client, line, MAX_LINE);
-                   Serial.println(line);
60                  if (line[0] == 'G' && line[1] == 'E' && line[2] == 'T')
-                     handle_command(line);
-                   if (!strcmp(line, "")) {
-                     client.println("HTTP/1.1 200 OK\n");
-                     break;
65                  }
-                 }
-               }
-               delay(1);
-               client.stop();
70           }
-         }
-       };
```

Nachdem wir alle benötigten Bibliotheken eingebunden haben, deklarieren wir die Klasse *InfraredProxy*. Wir definieren die Membervariable *_infrared_sender* zur Speicherung eines *IRsend*-Objekts, das wir benötigen, um infrarote Befehlscodes auszugeben.

In Zeile 8 definieren wir die Methode *read_line*, die eine vom Client gesendete Datenzeile liest. Eine Datenzeile endet entweder mit einem Zeilenvorschubzeilen (\n) oder mit einem Wagenrücklauf, gefolgt von einem Zeilenvorschub (\r\n). *read_line* erwartet das *Client*-Ethernetobjekt, von dem es Daten lesen soll, einen Zeichenpuffer zum Ablegen der Daten (*buffer*) und die Höchstlänge des

Zeichenpuffers (*buffer_length*). Die Methode ignoriert alle Zeilenvorschub- und Wagenrücklaufzeichen und setzt das letzte Zeichen einer Zeile auf \0, so dass der zu füllende Puffer stets ein mit NULL beendeter String ist.

Die nächste Methode (*send_ir_data*) beginnt in Zeile 20 und gibt einen Infrarotbefehl aus, der durch den Protokolltyp (*protocol*), die Länge des Codes in Bits (*bits*) und den zu sendenden Codewert (*value*) bestimmt ist. Abhängig vom Protokollnamen delegiert die Methode die gesamte Arbeit an die *IRsend*-Instanz.

handle_command analysiert den URL der HTTP-Anforderung und implementiert damit einen der schwierigsten Aspekte des Infrarotproxys. Um zu verstehen, was diese Methode macht, müssen wir wissen, wie HTTP-Anforderungen funktionieren. Wenn Sie in der Adressleiste Ihres Browsers einen URL wie *http://192.168.2.42/NEC/32/2011283550* eingeben, sendet der Browser eine HTTP-Anforderung, die wie folgt aussieht:

```
GET /NEC/32/2011283550 HTTP/1.1
host: 192.168.2.42
```

Die erste Zeile ist eine *GET*-Anforderung. *handle_command* erwartet einen String, der eine solche Anforderung enthält. Die Methode entnimmt alle Informationen, die in dem Pfad enthalten sind (*/NEC/32/2011283550*), und verwendet sie, um ein Infrarotsignal zu geben. Die Analyse dieser Informationen ist ein bisschen knifflig, aber mit der C-Funktion *strsep* nicht allzu schwierig. Diese Funktion trennt Strings auf, die durch bestimme Zeichen aufgeteilt sind. Sie erwartet einen String, der mehrere Teilstrings enthält, und einen String mit sämtlichen Trennzeichen. *strsep* ersetzt das erste Vorkommen jedes Zeichens aus dem Trennzeichenstring durch das Zeichen *\0* und gibt einen Zeiger auf den ursprünglichen String zurück. Vorher ersetzt es den Zeiger auf den String, den wir aufteilen möchten, durch einen Zeiger auf den ersten String.

Wir verwenden *strsep* in zwei verschiedenen Zusammenhängen. Im ersten Fall entnehmen wir den Pfad aus dem *GET*-Befehl, indem wir den String »GET« und den String »HTTP/1.1« entfernen, die beide durch ein Leerzeichen vom Pfad getrennt sind. Dies geschieht in den Zeilen 36 und 37. Wenn Sie beispielsweise den URL *http://192.168.2.42/NEC/32/2011283550* an *handle_command* übergeben, enthält *path* schließlich */NEC/32/2011283550*.

Jetzt liegt uns ein String vor, der aus drei durch Schrägstriche getrennten Teilstrings besteht. Nun müssen Sie *strsep* erneut einsetzen. Wenn Ihnen genau klar ist, was in den Zeilen 40 bis 43 geschieht, können Sie sich einer Vertrautheit sowohl mit C als auch mit der Funktion *strsep* rühmen. Schließlich enthält das Array *args* alle drei Pfadelemente. Wir können den Protokollnamen direkt an *send_ir_data* übergeben, müssen zuvor aber die Bitlänge und den Codewert in *int* bzw. *long* umwandeln. Für diese Umwandlung verwenden wir die Funktionen *atoi* und *atol*.

Jetzt haben wir alle Hilfsmethoden definiert, die wir brauchen, und müssen nur noch die öffentliche Schnittstelle der Klasse *InfraredProxy* implementieren.

Sie enthält nur die Methode *receive_from_sender*, die die Kernlogik der Klasse *InfraredProxy* implementiert. Diese Methode erwartet eine Instanz der Klasse *Server*, die in der Ethernet-Bibliothek definiert ist. Sie wartet darauf, dass ein Client mit der Methode *available* von *Server* Verbindung aufnimmt (Zeile 54). Wenn der Server mit einem Client verbunden ist, prüft sie, ob der Client neue Daten hat. Dies geschieht mit der *Client*-Methode *available* in Zeile 57.

receive_from_server liest die vom Client gesendeten Daten zeilenweise durch einen Aufruf von *read_line*. Zur Fehlersuche gibt die Methode jede Zeile am seriellen Port aus. Außerdem prüft sie jede Zeile darauf, ob sie mit *GET* beginnt. Wenn ja, ruft sie *handle_command* auf, anderenfalls prüft sie, ob die Zeile leer ist, da alle HTTP-Nachrichten mit einer leeren Zeile abgeschlossen werden. In diesem Fall sendet *receive_from_server* eine *OK*-Antwort zurück, wartet eine Millisekunde lang, um dem Client Zeit zur Verarbeitung der Antwort zu geben, und trennt dann mit dem Aufruf von *stop* die Verbindung zum Client.

Das war zugegebenermaßen sehr viel Code, aber der Erfolg ist der Mühe wert. Die Verwendung von *InfraredProxy* ist jetzt ganz einfach:

RemoteControl/InfraredProxy/InfraredProxy.pde

```
const unsigned int PROXY_PORT = 80;
const unsigned int BAUD_RATE = 9600;

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 2, 42 };

Server server(PROXY_PORT);
InfraredProxy ir_proxy;

void setup() {
  Serial.begin(BAUD_RATE);
  Ethernet.begin(mac, ip);
  server.begin();
}

void loop() {
  ir_proxy.receive_from_server(server);
}
```

Wie üblich geben wir die MAC- und die IP-Adresse an, die wir verwenden möchten. Anschließend definieren wir ein *Server*-Objekt und übergeben ihm den Port, den es abhören soll, nämlich 80 (den HTTP-Standard-Port). Außerdem initialisieren wir ein neues *InfraredProxy*-Objekt.

In der Methode *setup* initialisieren wir den seriellen Port zur Fehlersuche. Außerdem initialisieren wir den Ethernet-Shield und rufen die *begin*-Methode von *Server* auf, um den Listener unseres Servers zu starten. In *loop* rufen wir lediglich die Methode *receive_from_server* von *InfraredProxy* auf und übergeben ihr unsere *Server*-Instanz.

Nun wollen wir den Code endlich testen. Befestigen Sie den Ethernet-Shield an Arduino und die Infrarot-LED-Schaltung am Shield. Konfigurieren Sie die MAC- und die IP-Adresse, kompilieren Sie den Code und laden Sie ihn zu Arduino hoch. Öffnen Sie im Webbrowser den URL *http://192.168.2.42/NEC/32/2011283550* (passen Sie den URL an Ihre lokalen Einstellungen an!) und beobachten Sie, was mit Ihrem Mac oder dem Gerät geschieht, das Sie steuern möchten. (In Abbildung 9–9 sehen Sie die typische Ausgabe des Infrarotproxys auf dem seriellen Monitor.)

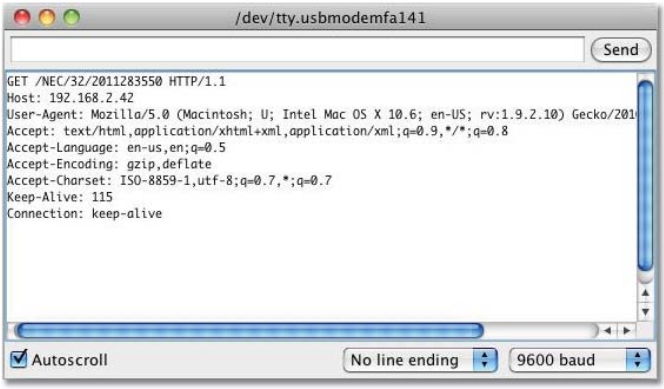

Abb. 9–9 *Zugriff auf den Infrarotproxy mit Firefox*

Auch wenn wir nur ein Minimum an Hardware verwendet haben (eine billige und einfache Infrarot-LED), sind die Projekte dieses Kapitels doch sehr nützlich und ziemlich anspruchsvoll, zumindest aus dem Blickwinkel der Softwareentwicklung. Wir können jetzt nicht nur jedes Gerät steuern, das Infrarotsignale versteht, sondern dazu auch den seriellen Port eines Computers und sogar einen Webbrowser heranziehen.

Es ist auch nicht mehr notwendig, Arduino mit dem USB-Anschluss Ihres Computers zu verbinden. Der Infrarotproxy beispielsweise nutzt den USB-Anschluss nur zur Stromversorgung. Wenn Sie einen Wechselstromadapter an Arduino anschließen, können Sie sich auch das USB-Kabel sparen.

Fernsteuerungen für alle möglichen Geräte

Bei allen Projekten in diesem Kapitel ging es um Geräte, die Sie bereits mit einer Infrarotfernbedienung steuern können. Es ist aber auch möglich, einen Infrarotempfänger an andere, bereits vorhandene oder neu konstruierte Geräte anzuschließen.

Im Prinzip könnten Sie also auch Ihren Kühlschrank und Ihre Mikrowelle mit einer Fernbedienung steuern. Wetten, dass Sie niemals an eine Fernsteuerung für einen Rasenmäher gedacht haben?[a]

a *http://www.instructables.com/id/Arduino-RC-Lawnmower/*

Zum ersten Mal haben wir jetzt Alltagsgegenstände mithilfe von Arduino gesteuert. Im nächsten Kapitel, in dem es um die Steuerung von Motoren geht, werden wir damit weitermachen.

9.7 Wenn es nicht funktioniert

In diesem Kapitel haben wir hauptsächlich LEDs und einen Ethernet-Shield verwendet, weshalb alle Ratschläge aus den Kapiteln 3, *Binäre Würfel*, und 8, *Netzwerkeln mit Arduino*, auch hier gelten.

Daneben müssen Sie aber noch auf einige andere Dinge aufpassen. Beispielsweise ist der Abstand zwischen der Infrarot-LED und dem Empfänger wichtig. Um auf der sicheren Seite zu sein, sollten Sie die LED nah am Empfänger platzieren. Sie sollte sich auch genau vor dem Empfänger befinden. Sorgen Sie auch dafür, dass es nicht zu viel Umgebungslicht gibt, das das Infrarotsignal stören könnte.

Zur Fehlersuche ist es sinnvoll, statt einer LED mit unsichtbarem Infrarotlicht eine normale LED zu verwenden. Dadurch können Sie sehen, ob Ihre Schaltung prinzipiell funktioniert.

Wenn Sie versuchen, einen Mac zu steuern, sollten Sie alle anderen Fernbedienungen im Sicherheitsbereich des Systemeinstellungsmenüs auf dem Mac deaktivieren.

Es kann auch sein, dass Ihr Gerät ein Protokoll verwendet, das von der Bibliothek IRremote nicht unterstützt wird. In diesem Fall müssen Sie es hinzufügen.. Das kann knifflig sein, aber da es sich bei IRremote um ein Open Source-Produkt handelt, ist es zumindest möglich.

9.8 Übungen

- Bauen Sie einen Emulator für eine Fernbedienung in Ihrem Haushalt. Machen Sie die Befehle über den seriellen Port und über Ethernet verfügbar.
- Steuern Sie Ardunio statt über einen seriellen Monitor oder einen Webbrowser mit einem Nintendo Nunchuk. Beispielsweise können Sie die Lautstärke Ihres Fernsehers regeln, indem Sie den analogen Hebel nach oben oder unten drücken, oder die Kanäle wechseln, indem Sie ihn nach links oder rechts bewegen.
- Entwickeln Sie eine echte Universalfernbedienung auf der Grundlage von Arduino. Suchen Sie nach einem Touchscreen, einer Tastatur, einem SD-Karten-Shield und einem Bluetooth-Modul. Ich wette, Sie hätten es niemals für möglich gehalten, dass Sie ein solches Gerät bauen könnten – aber Sie wissen jetzt alles, was Sie dazu wissen müssen!