Arduino verstehen und anwenden

Teil 9: WLAN und Micro-Server-Technik

Nachdem im letzen Beitrag zu dieser Artikelserie der Anschluss des Arduino an das lokale, drahtgebundene Heimnetzwerk im Vordergrund stand, geht es nun noch einen Schritt weiter. Auf die Kabelverbindung wird verzichtet und der Arduino wird drahtlos in ein WLAN (Wireless Local Area Network) eingebunden.

Natürlich muss dazu der eingesetzte Router über eine geeignete WLAN-Funktion (LAN 802.11b/g-Standard) verfügen. Diese ist aber mittlerweile bei allen aktuellen und gängigen Routern vorhanden.

Durch die Unabhängigkeit von einem LAN-Kabel erhält der Arduino eine bislang ungekannte Freiheit. Anstelle eines Netzwerkanschlusses ist nun nur noch eine Steckdose für die Spannungsversorgung erforderlich. Dies erweitert die Positionierungsmöglichkeiten in einem klassischen Haushalt ganz erheblich, da hier üblicherweise in jedem Raum mehrere Steckdosen zu finden sind. Ein Netzwerkanschluss ist dagegen, wenn überhaupt, häufig nur einmal in einem Zimmer vorhanden.

Die nahezu grenzenlose Freiheit erhält man, wenn der Arduino mit Batterien oder Akkus versorgt wird.

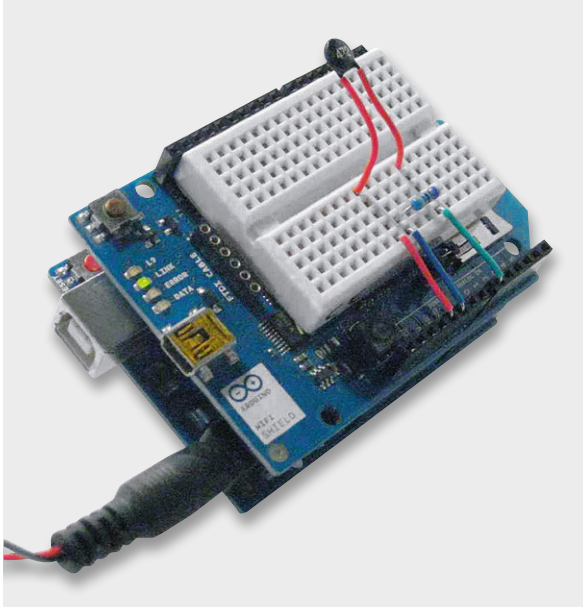

Bild 1: Arduino mit aufgestecktem Wi-Fi-Shield und Proto-Board

Eine solche Einheit kann überall in Haus, Keller oder Garten positioniert werden. Je nach Reichweite des WLAN kann so ein Bereich von bis zu 30 m oder mehr rund um den aktiven WLAN-Router abgedeckt werden. Von den vielen Möglichkeiten, die ein Arduino mit WLAN-Anschluss bietet, wird sich dieser Artikel auf die folgenden Themen konzentrieren:

1. Inbetriebnahme eines Wi-Fi-Shields
2. Funktionen und Anwendungen der Wi-Fi-Bibliothek
3. Der Arduino im WLAN
4. Drahtlose Übertragung von Sensordaten

Oft führt der Begriff „Wi-Fi", der auch bei drahtlosen Arduino-Anwendungen häufig auftaucht (z. B. „Wi-Fi-Bibliothek"), zu Verwirrungen und Missverständnissen. Wi-Fi ist ein Kunstwort, das oftmals als Abkürzung für „Wireless Fidelity" aufgefasst wird. Hier kommt die Anlehnung an den Hi-Fi-Begriff der Audiotechnik zum Ausdruck. Mit einem Wi-Fi-Netzwerk soll so ein Funkstandard für höchste Ansprüche assoziiert werden. Meist wird Wi-Fi aber einfach als Synonym für WLAN benutzt, obwohl genau genommen der Begriff WLAN das Funknetzwerk, Wi-Fi dagegen den Funkstandard bezeichnet.

Das Arduino Wi-Fi Shield

Im Gegensatz zum Ethernet-Anschluss hat man beim WLAN nicht die Auswahl zwischen einem Arduino mit integriertem WLAN-Interface und einem WLAN-Shield. Will man einen Arduino in ein drahtloses Netzwerk integrieren, wird immer ein klassischer Arduino und ein zusätzliches WLAN-Shield benötigt.

Bild 1 zeigt die fertig montierte Kombination. Zusätzlich wurde hier noch ein kleines Proto-Board eingesetzt, welches einen Temperatursensor trägt.

Beim Aufstecken des Shields auf das Arduino-Board ist zu beachten, dass nicht nur die Pins der oberen und unteren Stiftleiste verbunden werden müssen, sondern zusätzlich auch noch die sechs Anschlüsse der ICSP-Schnittstelle.

Diese Kombination bietet die folgenden Leistungsmerkmale:

- Betriebsspannung: 5 V (Versorgung des Wi-Fi-Shields erfolgt über den Arduino)
- WLAN-Protokoll: 802.11b/g
- Verschlüsselung: WEP und WPA2

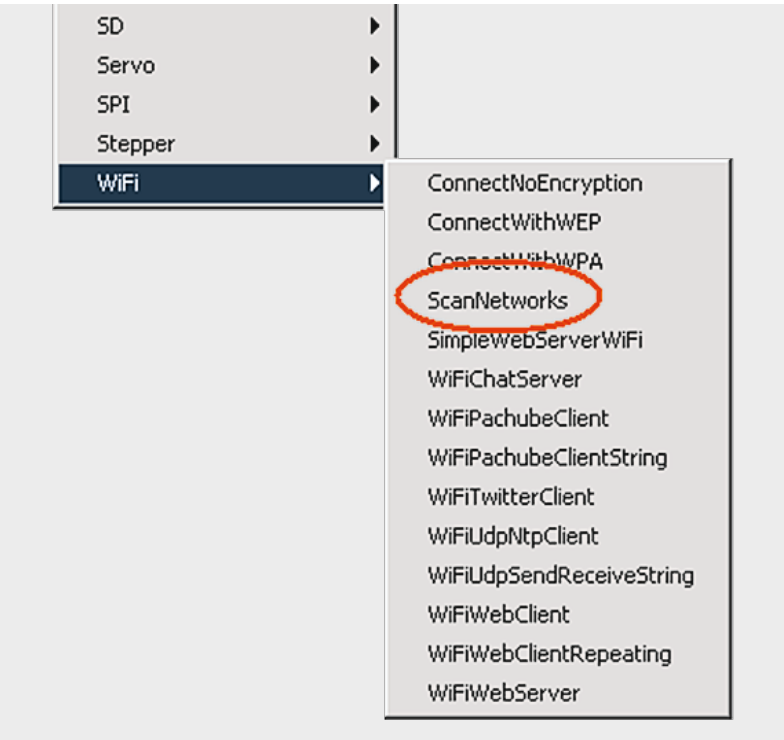

Bild 2: Das Menü zur Wi-Fi-Library

Zusätzlich verfügt das Wi-Fi-Shield genau wie die Ethernet-Versionen des Arduino über einen Onboard-microSD-Karten-Slot. Eine separate Mini-USB-Buchse erlaubt die Aktualisierung der Firmware auf der Wi-Fi-Karte.

Das Wi-Fi-Shield enthält einen eigenen ATmega-32UC3-Chip. Dieser beinhaltet einen IP-Stack, welcher sowohl TCP als auch UDP unterstützt. (Nähere Details zu den Netzwerkprotokollen wurden bereits im letzten Artikel dargelegt.)

Die Anschlüsse des Arduino sind über die Buchsenleisten des Wi-Fi-Shields durchgeführt, so dass die meisten Pins für eigene Anwendungen weiterhin zur Verfügung stehen. Im Bedarfsfall können auch weitere Shields auf das Wi-Fi-Board aufgesteckt werden.

Auf den im Wi-Fi-Shield integrierten microSD-Karten-Steckplatz kann über die normale SD-Bibliothek zugegriffen werden. Auf diese Weise ist es möglich, auch umfangreichere Webseiten zu erstellen und auf einer entsprechend großen SD-Karte abzulegen.

Da der Arduino sowohl mit dem Prozessor des Wi-Fi-Shields als auch mit der integrierten SD-Karte über den SPI-Bus kommuniziert, muss auch der ICSP-Stecker elektrisch mit dem Shield verbunden werden. Über den I/O-Pin Nummer 10 wird der Chip-Select des Wi-Fi-Controllers HDG104, über Pin 4 der Select für die SD-Karte gesteuert. Diese beiden Pins stehen also nicht mehr für externe Aufgaben zur Verfügung.

Der Digitalpin 7 wird für das Handshake zwischen dem Wi-Fi-Shield und dem Arduino eingesetzt und sollte daher ebenfalls nicht für andere Aufgaben verwendet werden.

Neben dem Zugriff auf offene und unverschlüsselte drahtlose Netzwerke ermöglicht das Wi-Fi-Shield auch die Einbindung in WLANs mit WPA2- oder WEP-Verschlüsselung.

Für die Anzeige der Betriebszustände verfügt das Wi-Fi-Shield über 4 Status-LEDs:

- L9 (gelb): Diese LED ist mit Digitalpin 9 verbunden
- LINK (grün): Diese LED dient als Indikator für die Netzwerkverbindung
- ERROR (rot): Die rote LED zeigt Kommunikationsfehler an
- DATA (blau): Die blaue Data-LED blinkt, wenn Datenpakete gesendet oder empfangen werden

Die Wi-Fi-Library

Wie im Arduino-Umfeld üblich, wird auch zum Wi-Fi-Shield eine umfangreiche Bibliothek mit mehreren Beispielen zur Verfügung gestellt. Die Bibliothek gehört bei den aktuellen IDE-Versionen bereits mit zur Standardausstattung.

Bild 2 zeigt, wie die Wi-Fi-Biliothek aufgerufen wird. Eine sehr interessante und durchaus auch nützliche Anwendung aus der Standard-Beispielsammlung ist der Sketch „ScanNetworks". Wie üblich kann der Sketch über die Menüauswahl geladen werden (siehe Bild 2).

Nach dem Start des Sketches und des seriellen Monitors wird eine Übersicht über alle lokal verfügbaren Drahtlosnetze angezeigt. Neben dem Funktionstest des Shields kann man mit dieser Anwendung auch die Signalstärken in der Umgebung des WLAN-Routers ausmessen. Mit einem etwas längeren USB-Kabel oder mit einem Laptop können so die günstigsten Positionierungen für WLAN-Elemente bestimmt werden.

Wie entsprechende Messungen zeigen, liegt die Empfindlichkeit und Reichweite des Arduino-Wi-Fi-Shields im mittleren Bereich. Mit einem modernen, mit integriertem WLAN ausgerüsteten Laptop erhält man deutlich bessere Reichweiten. Dies ist insbesondere dadurch begründet, dass in derartigen Laptops großflächige Empfangsantennen integriert sind.

Smartphones dagegen sind von der WLAN-Empfangsleistung oftmals nicht deutlich besser als das WLAN-Shield. Kleine und preisgünstige Mini-WLAN-USB-Adapter erreichen dagegen sogar oftmals nicht die Reichweite des Shields. Dies ergibt sich natürlich auch aus der Tatsache, dass in die kleine Bauform der Sticks keine effektiven Antennen integriert werden können.

Bild 3 zeigt die Ausgabe von Signalfeldstärken im seriellen Monitor der Arduino-IDE. Wie ebenfalls zu erkennen ist, werden alle am Standort verfügbaren WLAN-Netze angezeigt. So eignet sich der Arduino mit Wi-Fi-Shield also auch für den Einsatz als Netzwerkscanner.

Hinweis:
Wie bereits im Beitrag zur LAN-Einbindung des Arduino wurden auch hier wieder MAC- und Ethernet-Adressen sowie Passwörter etc. aus Datenschutzgründen ausgeblendet.

Die Signalstärke wird hier in dBm angegeben. Diese Einheit steht für Dezibel Milliwatt und bezeichnet den Leistungspegel in Dezibel, bezogen auf 1 mW Leistung. Zwischen der Leistung (P) in Milliwatt (mW) und dem dBm-Wert besteht folgender Zusammenhang:

$$P/dBm = 10*\log(P/1\ mW)$$

Ein Milliwatt entspricht also 0 dBm. Größere Leistungswerte haben positive, kleinere negative dBm-Werte. Negativen dBm-Werten sind damit Leistungen zugeordnet, die kleiner sind als 1 mW.

So entsprechen:

100 µW	-10 dBm
10 µW	-20 dBm
1 µW	-30 dBm
0,1 µW	-40 dBm
0,01 µW	-50 dBm
1 nW	-60 dBm

Die in Bild 3 angegebenen Leistungen liegen also erwartungsgemäß deutlich unterhalb von einem Milliwatt, sind aber für einen brauchbaren WLAN-Empfang durchaus noch ausreichend.

Hat man so einen Standort mit gutem WLAN-Empfang gefunden, kann man den nächsten Schritt in Angriff nehmen und den Arduino in das drahtlose Netzwerk einbinden.

Der Arduino im WLAN

Für den Aufbau einer aktiven WLAN-Verbindung kann man den Arduino zunächst als Wi-Fi-Webserver konfigurieren. Ein entsprechender Sketch (WiFiWebServer.ino) findet sich in der Wi-Fi-Library.

Bevor dieser Sketch auf den Arduino geladen wird, müssen noch einige Konfigurationen ausgeführt werden.

Ein Heimnetzwerk sollte keinesfalls ohne Verschlüsselung betrieben werden. Eine weit verbreitete Verschlüsselungsmethode ist WPA2. Das Wi-Fi-Shield ist in der Lage, über diesen Standard zu kommunizieren. Im Sketch müssen dafür zwei Angaben gemacht werden:

1. In char ssid[] = "yourNetwork"; muss die Netzwerk-SSID eingetragen werden. Angaben dazu finden sich im Manual zum verwendeten Router.
2. Das Passwort wird dem Arduino unter
   char pass[] = "secretPassword";
   übergeben.

Dann kann der entsprechende Sketch zum Arduino übertragen werden. An dieser Stelle ist jedoch ein wichtiger Hinweis angebracht:

Hinweis
Die WLAN-Shields arbeiten nicht mit allen IDE-Versionen. Die besten Ergebnisse wurden mit der IDE-Version 1.0.2 erzielt. Insbesondere mit den aktuellen Versionen 1.0.5 bzw. 1.0.6 scheint es mit dem Wi-Fi-Shield Probleme zu geben. Dazu gibt es Beiträge in verschiedenen Foren und auch eigene Versuche haben dies bestätigt.

Wurden die notwendigen Angaben zur SSID und zum Passwort korrekt eingegeben, kann der Sketch, vorzugsweise mit der Arduino-IDE 1.0.2, auf den Arduino übertragen werden.

Ist der Sketch erfolgreich geladen, versucht das Wi-Fi-Shield Verbindung zum angegebenen WLAN-Router aufzunehmen. Sobald die Verbindung steht, leuchtet die grüne „LINK"-LED. Falls der Versuch fehlschlägt, wird die rote „ERROR"-LED aktiviert. In diesem Fall sollte man die Angaben im Sketch nochmals genau

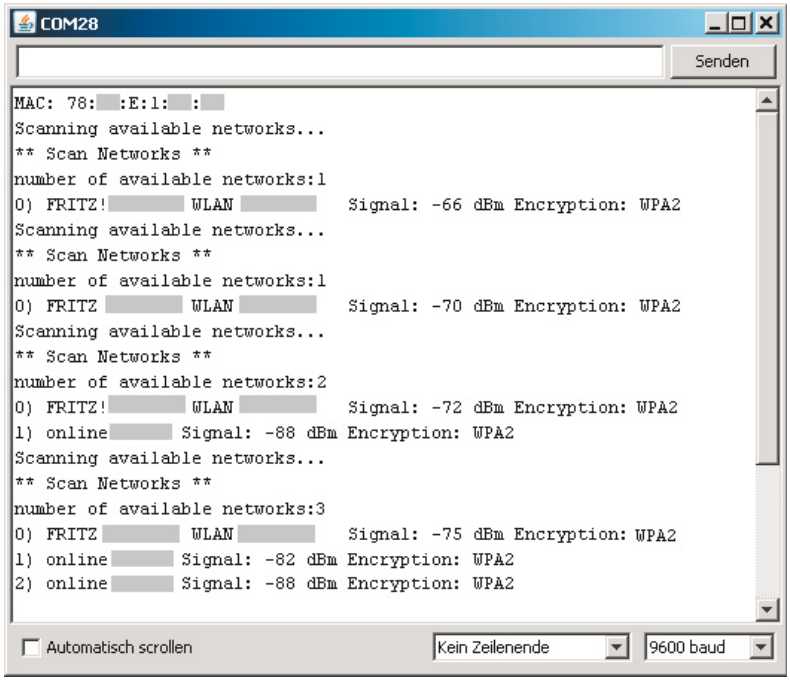

Bild 3: WLAN-Netzwerkscan mit dem Wi-Fi-Shield

kontrollieren. Zudem ist auch zu bedenken, dass, wie bereits erläutert, die Empfangsleistung des Wi-Fi-Shields nicht ganz so optimal ist wie die eines Tablets oder eines Laptops mit integrierter WLAN-Einheit. Hier kann es also hilfreich sein, weitere Tests in direkter Umgebung des WLAN-Routers durchzuführen.

War die Verbindungsaufnahme schließlich erfolgreich (grüne LED leuchtet permanent), kann man in der Arduino-IDE den seriellen Monitor öffnen (Bild 4).

Hier wird nun die aktive Kommunikation mit dem Drahtlos-Netzwerk nochmals bestätigt. Außerdem wird die IP-Adresse angegeben, welche über das Netzwerk dem Arduino zugeordnet wurde. Das folgende Bild zeigt die Ausgabe (zur Wahrung des Datenschutzes wurden wieder einige Angaben ausgegraut).

Die an den seriellen Monitor ausgegebene IP-Adresse kann man nun in einen Browser (z. B. Explorer, Chrome oder Firefox) eingeben und so Kontakt zum Arduino aufnehmen. Details dazu wurden bereits im letzten Beitrag dieser Serie erläutert. An dieser Stelle macht es keinen Unterschied, ob der Arduino über LAN oder WLAN eingebunden ist.

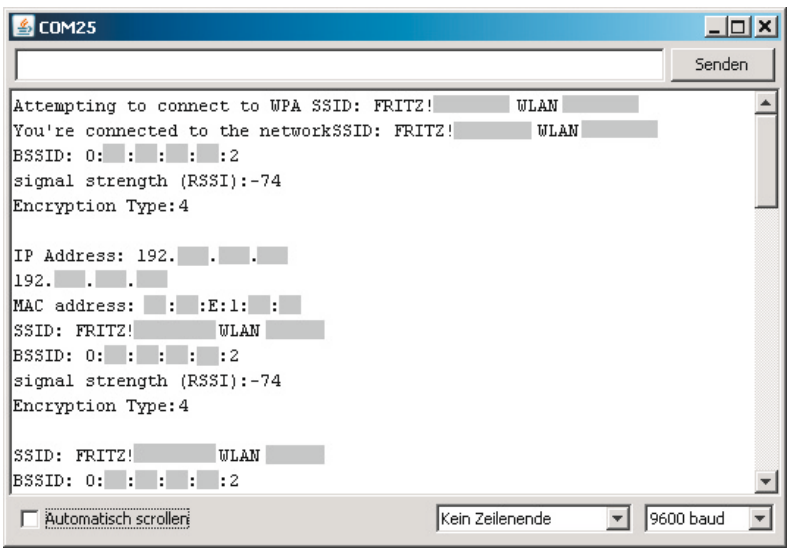

Bild 4: Verbindung zum WLAN-Router ist aufgebaut.

Wenn alles korrekt funktioniert, sollte im Browser-Fenster folgende Ausgabe erscheinen:

```
analog input 0 is 411
analog input 1 is 398
analog input 2 is 207
analog input 3 is 173
analog input 4 is 297
analog input 5 is 350
```

Die Werte der Analogeingänge sind zufällig und ändern sich in rascher Folge, da die offenen Eingänge lediglich elektromagnetische Störungen einfangen (siehe dazu auch Teil 6 der Artikelserie „Sensortechnik und Messwerterfassung" im ELVjournal 5/2014).

Der Enviro-Server

Will man auch sinnvollere Daten als elektromagnetische Störungen empfangen, dann kann man den Arduino mit WLAN-Shield zu einem Server für Umweltmessdaten, einem sogenannten „Environmental-" oder kurz „Enviro"-Server, erweitern. Die Grundlagen hierfür sind ebenfalls bereits aus dem Artikel zu Sensortechnik und Messwerterfassung bekannt. Will man beispielsweise Umgebungs- oder Raumtemperaturen messen und via WLAN übertragen, kann man dazu einen NTC-Temperatursensor einsetzen. Bild 5 zeigt eine Schaltung dazu.

Zusätzlich kann man nun den Arduino mit einem Akku betreiben. Da der Arduino UNO bereits einen Spannungsregler an Board hat, kann man den Akku direkt an die Hohlbuchse des Controllerboards anschließen (Bild 6). Man erhält so einen vollkommen autarken Micro-Server, da nunmehr weder Daten noch Stromversorgungsleitungen erforderlich sind. Für die Akkuversorgung kommen entweder NiMH- oder Li-Ionen-Zellen in Frage. Die durchschnittliche Stromaufnahme des Arduino mit WLAN-Shield beträgt ca. 120 mA bei 8,4 V (z. B 2x Li-Ion- oder 7x NiMH-Zellen). Damit lassen sich problemlos Laufzeiten von mehreren Stunden bis zu einigen Tagen erreichen. Ein Sketch für den Betrieb des Enviro-Servers kann dann so aussehen:

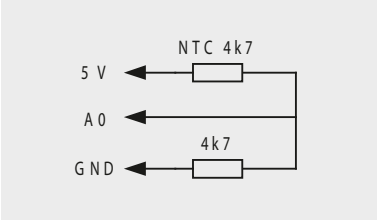

Bild 5: Schaltung für einen NTC-Temperatursensor am Wi-Fi-Shield

```
// web  Server using WiFi with WPA encryption - temperature sensor using NTC

int ADC0=0, int ADC_value;            // select and init ADC channel
float temp, cal=6.8, float offset=310;        // calibration data for NTC 4k7

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

char ssid[] = "WLAN!Box xxxx";  // network SSID
char pass[] = "xyz";                    // network password
int keyIndex = 0;                                 // your network key Index number
int status = WL_IDLE_STATUS;
WiFiServer server(80);

void setup()
{ Serial.begin(9600);
  if (WiFi.status() == WL_NO_SHIELD)
  { Serial.println("WiFi shield not present"); while(true);  }
  while ( status != WL_CONNECTED)
  { Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);  delay(10000);
  }
  server.begin(); printWifiStatus();
}

void loop()
{ WiFiClient client = server.available();
  if (client)
  { Serial.println("new client");
    boolean currentLineIsBlank = true;
    while (client.connected())
    {   if (client.available())
        { char c = client.read();
          Serial.write(c);
          if (c == '\n' && currentLineIsBlank)
          { client.println("HTTP/1.1 200 OK"); client.println("Content-Type: text/html");
                client.println("Connnection: close"); client.println();
                client.println("<!DOCTYPE HTML>"); client.println("<html>");
                client.println("<meta http-equiv=\"refresh\" content=\"5\">");
                ADC_value=analogRead(ADC0);
                client.print("Temperature @ server site: ");  temp=((ADC_value-offset)/cal);
                       client.print(temp, 1); client.print(" C"); client.println("<br />");
                       client.println("</html>");
                break;
          }
        if (c == '\n') {  currentLineIsBlank = true;     }
        else if (c != '\r') {  currentLineIsBlank = false;         }
      }
    }
    delay(1);  client.stop();  Serial.println("client disconnected");
  }
}

void printWifiStatus()
{ Serial.print("SSID: "); Serial.println(WiFi.SSID());
   IPAddress ip = WiFi.localIP(); Serial.print("IP Address: "); Serial.println(ip);
   long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):"); Serial.print(rssi);  Serial.println(" dBm");
}
```

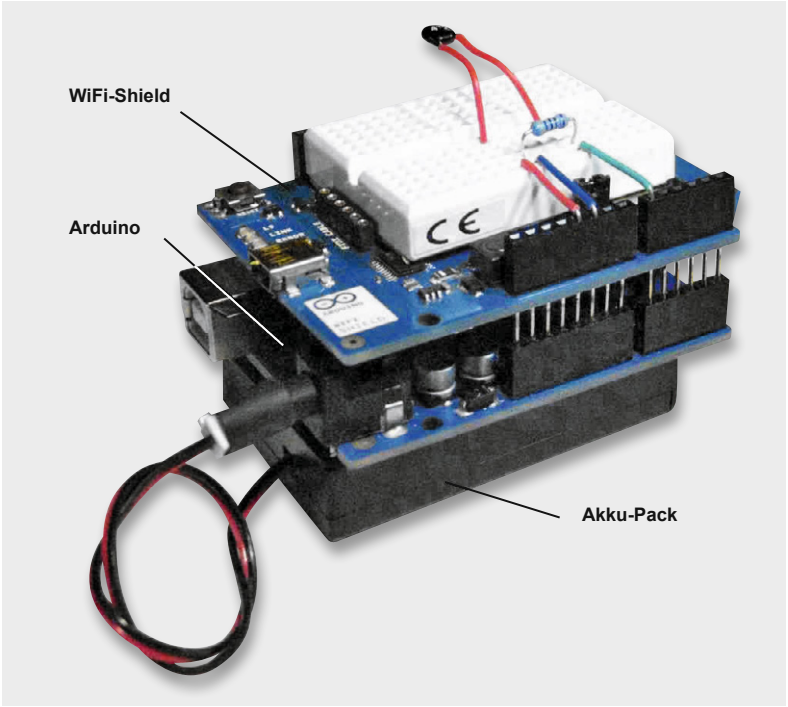

Bild 6: Autonomer WLAN-Server mit Akku-Pack

Bild 7: Anzeige der Temperatur in einem Browser

Nachdem der Sketch auf den Arduino übertragen wurde, kann der Enviro-Server über einen beliebigen Browser aufgerufen werden, wie in Bild 7 zu sehen.

Natürlich ist es nun auch möglich, den Server beispielsweise über ein WLAN-fähiges Smartphone aufzurufen. Hierzu ist wieder lediglich ein beliebiger Webbrowser zu starten. Dann muss nur noch die korrekte IP-Adresse angegeben werden, und der Enviro-Server kann drahtlos von jedem beliebigen Punkt innerhalb der WLAN-Reichweite ausgelesen werden. Bild 8 zeigt die entsprechende Darstellung auf einem Android-Smartphone.

Ausblick

Nach der Einbindung des Arduino in ein drahtgebundenes LAN im letzten Teil der Artikelserie wurde in diesem Beitrag die drahtlose WLAN-Technologie genutzt, um einen autarken, Arduino-basierten Micro-Server abzufragen.

Damit ist der Ausflug in den Bereich der Netzwerktechnik zunächst abgeschlossen. Der nächste Teil der Artikelserie wird sich um das Thema „Interrupts und Polling" drehen. Hier kommen dann bereits recht fortgeschrittene Programmiertechniken zum Einsatz, die es beispielsweise erlauben, nahezu verzögerungsfrei auf unvorhergesehene Ereignisse zu reagieren. Typische Anwendungen der Interrupt-Technologie reichen dabei von komfortablen manuellen Eingabemethoden bis hin zur Alarmauslösung in Gefahrensituationen. ELV

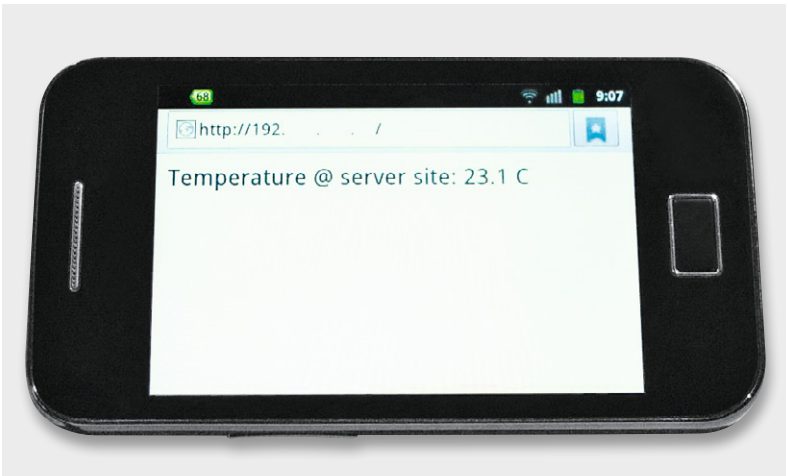

Bild 8: Ablesen der Server-Daten auf einem Mobilgerät

Weitere Infos:

- G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag 2012, Best.-Nr. J8-10 94 45, € 39,80
- Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr. J8-10 20 44, € 99,–
- Grundlagen zur elektronischen Schaltungstechnik finden sich in der E-Book-Reihe „Elektronik!" (www.amazon.de/dp/B000XNCB02)
- Buch „AVR-Mikrocontroller in C programmieren", Franzis-Verlag 2012, Best.-Nr. J8-09 73 52, € 39,95

Preisstellung Februar 2015 – aktuelle Preise im Web-Shop

Empfohlene Produkte/ Bauteile:	Best.-Nr.	Preis
Arduino-Wi-Fi-Shield	J8-11 02 51	€ 89,95
Arduino UNO	J8-10 29 70	€ 27,95
Mikrocontroller Online-Kurs	J8-10 20 44	€ 99,–

Alle Arduino-Produkte wie Mikrocontroller-Platinen, Shields, Fachbücher und Zubehör finden Sie unter: www.arduino.elv.de