Das Beaglebone-Black-System ist mit einer schwarzen Platine aufgebaut und hat einen Vorgänger, den *Beaglebone* mit weißer Platine, der demgegenüber weniger gut ausgestattet und fast doppelt so teuer ist. Mit diesem Modell wurde der Steckplatz für die Capes etabliert, wovon sich maximal vier Stück übereinander stecken lassen. Zu beachten ist, ob die jeweilige Cape-Platine auch mit dem Beaglebone Black kompatibel ist, denn es haben sich einige Signalveränderungen gegenüber dem Vorgänger ergeben.

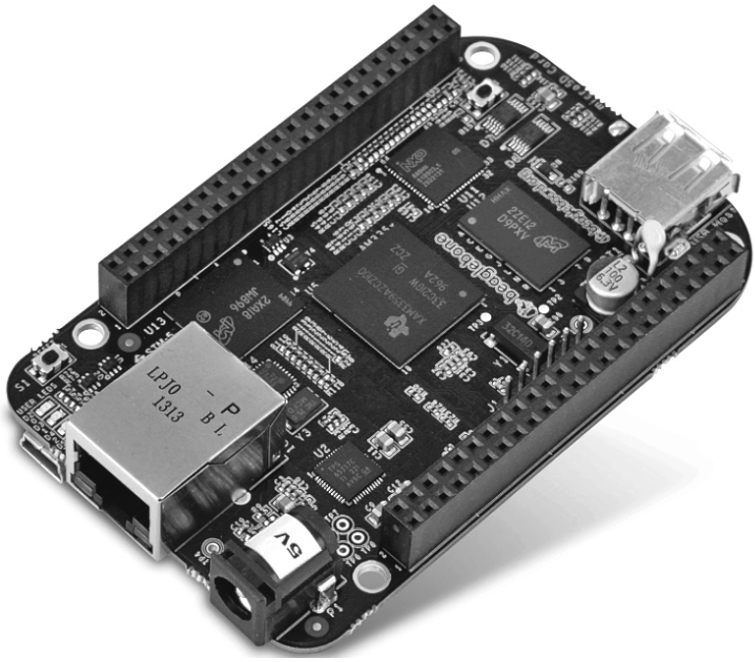

Abb. 3–15 *Der Beaglebone Black bietet sich als Alternative zum Raspberry Pi an.*

Durch die standardmäßig vorhandene Entwicklungsumgebung kann mit dem Experimentieren und Programmieren eigener Applikationen, etwa für den Einsatz der GPIO-Ports, sofort begonnen werden. Cloud9 startet automatisch beim Boot, und der Zugriff erfolgt mithilfe eines Webbrowsers auf den Webserver der IDE. Standardmäßig wird hierfür die Adresse 3000 im Browser Epiphany verwendet, was ebenfalls mit den beiden anderen standardmäßig vorhandenen Browsern Chrome und Firefox durchführbar ist.

Cloud9 ist für Node.js vorkonfiguriert, das eine JavaScript-basierte Plattform für Serveranwendungen darstellt. Die dazugehörige Laufzeitumgebung ist recht ressourcenschonend und wurde ursprünglich von Google für den Chrome-Browser entwickelt. Dazu gehört eine spezielle Bibliothek (Bonescript Library), die

einen einfachen Zugriff auf die unterschiedlichen Hardware-Einheiten – ähnlich wie bei Arduino – ermöglicht. Im Gegensatz zum Arduino muss der ausführbare Programmcode beim Beaglebone Black nicht explizit auf die Platine geladen werden, denn er wird automatisch im Dateisystem gespeichert.

Als Beispiel ist im Folgenden der Node.js-Code angegeben, der notwendig ist, um die LED USR3 auf dem Board anzusteuern. Nachdem der entsprechende Anschluss als Ausgang definiert ist, wird die LED mithilfe einer Schleifenfunktion und einer Verzögerungszeit von jeweils einer Sekunde laufend ein- und ausgeschaltet.

```
require(´bonescript´);

ledPin = bone, USR3;

setup = function (){
        pinMode(ledPin, OUTPUT);
};

loop = function (){
        digitalWrite(ledPin, HIGH);
        delay (1000);

        digitalWrite(ledPin, LOW);
        delay (1000);

};
```

Die bevorzugte Art der Programmierung ist beim Beaglebone-Black-System der Einsatz von Node.js-JavaScript, was sich insbesondere für Einsteiger und kleinere Applikationen empfiehlt. Gleichwohl kann wie beim Raspberry Pi auch direkt aus dem Linux-System heraus auf GPIO-Anschlüsse zugegriffen werden, und dem Einsatz von Python und C mitsamt den passenden Bibliotheken steht auch hier nichts im Wege, was in Kapitel 7 ausgeführt wird.

Unter der Bezeichnung *Beagle Board* sind außerdem weitere ARM-Boards verfügbar, die quadratische Abmessungen aufweisen, etwa das Beagleboard xM, das zusätzlich mit einer RS232-Schnittstelle, einem S-Video-Ausgang und einer Kamera ausgestattet ist, dafür aber auch ca. 150 € kostet. Typischerweise wird auf den Beagle Boards ein OMAP-Typ (3530-SoC) von Texas Instruments eingesetzt. Außerdem werden eine ganze Reihe von ähnlich ausgestatten Boards angeboten, die teilweise auch untereinander kompatibel sind, wie beispielsweise die Pandaboards, die mit einem OMAP 4460-SoC arbeiten, der zwei Cortex-A8-Kerne enthält und ebenfalls von Texas Instruments stammt.

Die Quellen sind für die Beagleboards komplett offengelegt, wobei die einzelnen Bauelemente auch einzeln verfügbar und keine speziellen Entwicklungen für bestimmte Boards sind, wie es beispielsweise beim Raspberry Pi der Fall ist. Insbesondere für die Beagleboards, inklusive des Beaglebone Black, existiert eine recht aktive Community *(beagleboard.org)*. Hier sind genaue Informationen zu den Boards, Schaltpläne, Betriebssystem-Images (Ubuntu, Android) und auch Programmierbeispiele sowie komplette Projekte zu finden. Initiiert wurde die Beagleboard-Open-Source-Community vom internationalen Distributor Digi-Key aus den USA, während die offiziellen Raspberry-Pi-Distributoren die Konkurrenzfirmen Farnell (Element 14) und RS-Components sind.

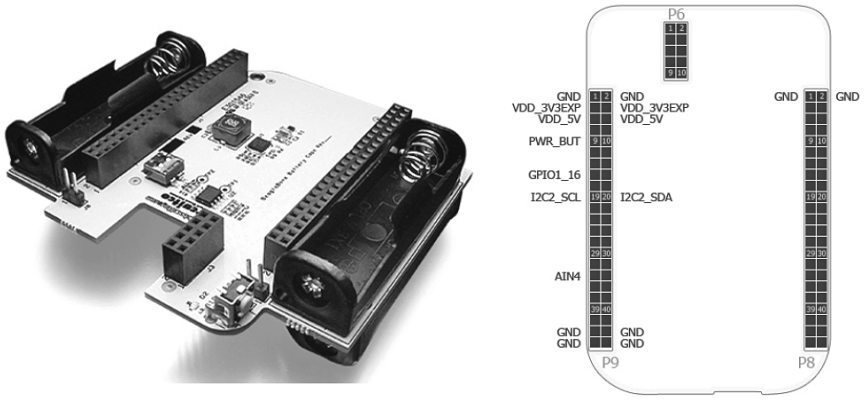

Abb. 3–16 *Mithilfe des Battery Capes kann die Spannungsversorgung über vier AA-Batterien erfolgen.*

3.3 mbed-Plattform

Unter der Bezeichnung *mbed* gibt es eine ganze Reihe von Cortex-M3- und Cortex-M0-Boards, die hauptsächlich Mikrocontroller der Firma NXP verwenden, wie die Typen LPC1768 (M3), LPC4088 (M3), LPC1347 (M3) und LPC11U24 (M0), LPC800 (M0), LPC1114FN28 (M0). Die Firma Freescale firmiert mit dem KL25Z (M0) ebenfalls unter der mbed-Plattform. Außerdem wird mit dem NXP LPC4088 auch ein Cortex M4 unter mbed geführt. Die Boards sind in verschiedenen Platinenausführungen und mit unterschiedlichen Peripherie-Chips aufgebaut. Besonders praktisch sind die Typen, die als Modul mit ihren 40 Pins in einen entsprechenden DIP-Sockel oder auch in ein Steckbrett passen. Der Typ LPC1114FN28 kombiniert das komplette System in einem 28-poligen DIP-Gehäuse.

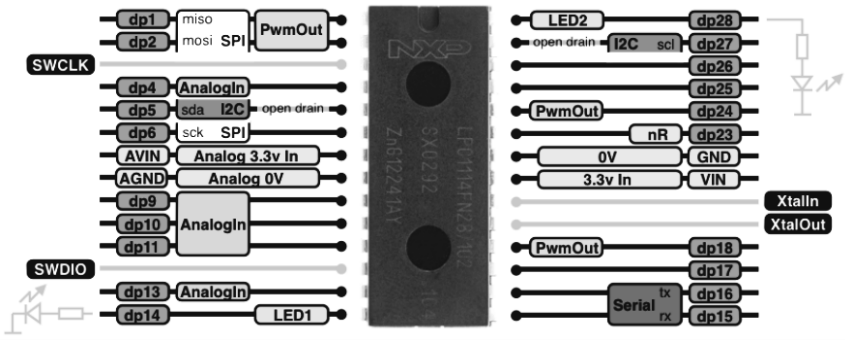

Abb. 3–17 *Ein Cortex M0 für die mbed-Plattform mit einer Vielzahl von Anschlussmöglichkeiten in einem 28-poligen DIP-Gehäuse eignet sich besonders für Test- und Hobby-Applikationen.*

Die mbed-Entwicklungsumgebung ist in sofern bemerkenswert, als sie nicht lokal auf einem PC, sondern online auf der Internetseite *mbed.org* beim jeweiligen Modell mit einem üblichen Browser ausgeführt wird. Dies setzt dort einen Account voraus, der kostenlos zur Verfügung gestellt wird. Das Programm wird als C/C++-Code im Webeditor eingegeben und der Online-Compiler/Linker generiert daraus ein lauffähiges Programm, das herunterzuladen und dann per USB-Verbindung auf das mbed-Board zu kopieren ist. Das heißt, der Controller ist damit programmiert. Dieses Verfahren ist für die professionelle Nutzung eher ungeeignet, weil die eigenen Programme auf einem fremden Webserver gespeichert werden, und Debugging-Funktionen gibt es auch nicht. Der mbed-Ansatz entspricht augenscheinlich dem Community-Prinzip, bei dem die Anwender einen regen Austausch untereinander pflegen wollen.

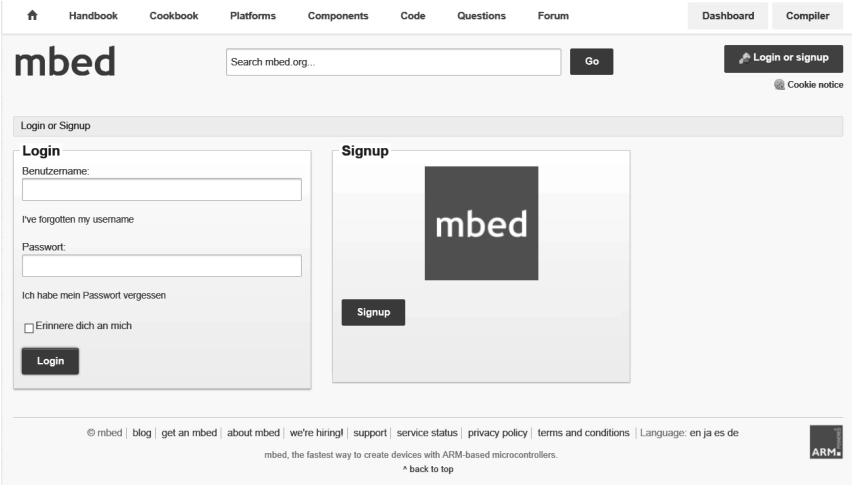

Abb. 3–18 *Ohne Login und Account kommt man mit einem mbed-System nicht weiter.*

Einer der verbreitetsten Typen der mbed-Plattform ist das Board mit dem LPC1768 (Cortex M3), das zu einem Preis von ca. 50 € erhältlich ist. Im Gegensatz zu den zuvor in diesem Kapitel erläuterten Boards (vgl. auch Abschnitt 2.5.3 über die Gecko-Familie) ist hierfür kein eigenes (Linux-)Betriebssystem verwendbar, weil dieser Mikrocontroller aufgrund einer fehlenden *Memory Management Unit* (MMU) keine virtuelle Speicherverwaltung unterstützt.

Die mbed-Systeme sind demnach explizit für Embedded Systems vorgesehen. Der LPC1768 wird mit 96 MHz getaktet und enthält 64 kByte RAM sowie 512 kByte Flash-Speicher, was für die meisten Applikationen ausreichend ist. Er verfügt über 26 digitale I/O-Leitungen, über sechs analoge Eingänge (12-Bit-A/D) und als Besonderheit gegenüber anderen Lösungen auch noch über einen analogen Ausgang (10-Bit-D/A). Als Schnittstellen sind USB, SPI, I^2C, wie bei den anderen Systemen auch, sowie drei UART-Interfaces und sogar noch ein CAN-Interface vorhanden.

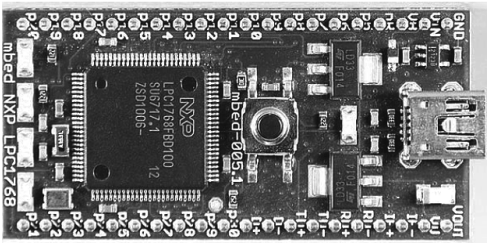

Abb. 3–19 *Das LPC1768-Board*

Beim erstmaligen Verbinden des mbed per USB-Kabel mit einem PC, aus dem auch der Strom für den Betrieb des Boards bezogen wird, erkennt das Betriebssystem (Windows) zunächst einen Mikrocontroller und daraufhin ein *USB-Massenspeichergerät*, das das mebed-Board in Form des Laufwerks MBED repräsentiert. Wenn die dort befindliche Datei mbed.htm aufgerufen wird, wird die Internet-Verbindung zur mbed-Webseite hergestellt, auf der zunächst der Login zu erfolgen hat. Nach dem Signup wird ein Hello-Word-Beispiel als erstes Versuchsprogramm vorgeschlagen, das durch Anklicken auf der Webseite auf dem mbed-Laufwerk des PCs gespeichert werden kann. Nach der Betätigung der Taste auf dem Board wird es dann in den Flash-Speicher geladen, was durch das kurze Blinken einer blauen LED erkennbar ist. Daraufhin wird das Programm automatisch gestartet, sodass die erste LED der vier Betriebs-LEDs fortwährend blinkt. Das Schreiben eigener Programme wird durch das Anklicken der Option *Compiler* gestartet. Dadurch wird ein Workspace für die Aufnahme des C-Codes eröffnet, der an dieser Stelle erstellt, editiert und kompiliert werden kann.

3.4 Gadgeteer

Zurzeit werden von allen möglichen Firmen und Herstellern verschiedenste Mikrocontroller- und Einplatinensysteme vorgestellt. Selbst Microsoft ist mit einer eigenen Lösung – Gadgeteer – dabei. Gadgeteer benötigt ein spezielles Board, das nicht von Microsoft selbst, sondern von verschiedenen Hardware-Herstellern angeboten werden soll. Momentan stehen ausschließlich Boards der Firma GHI Electronics (FEZ Spider Kit) zur Verfügung. Üblicherweise verwenden die Gadgeteer-Mainboards einen Prozessor mit ARM-Architektur; beim Spider Kit wird ein ARM 7 mit 72 MHz eingesetzt.

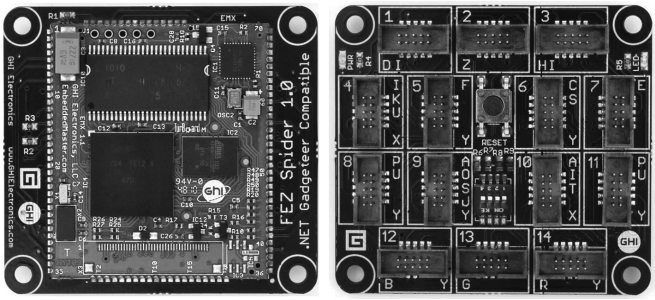

Abb. 3–20 *Das Gadgeteer-Mainboard von der Rück- und von der Vorderseite, wo fast nur Steckpfosten angebracht sind.*

Ein Gadgeteer-Mainboard kann mit verschiedenen Erweiterungs-Boards verbunden werden, die als *Sockets* bezeichnet werden. Während dies bei anderen Systemen über Platinen (Capes, Shields) erfolgt, die sich direkt auf das jeweilige Controller-Board stecken lassen, werden die Sockets über Flachbandkabel mit dem Gadgeteer-Board verbunden. Die hierfür vorgesehenen Steckpfosten (siehe Abbildung 3–20, rechts) sind jeweils 10-polig, und jeder der 14 Pfosten führt ganz bestimmte Signale: Pfosten 1 beispielsweise USB und I^2C, Pfosten 5 die Signale für ein SD-Karten-Interface, Pfosten E die Signale für ein Ethernet-Interface, Pfosten 10 drei analoge Eingangssignale, und die Pfosten 12 bis 14 sind für den Anschluss von Displays vorgesehen.

Die Gadgeteer-Programme werden in C# mithilfe des Microsoft Visual Studios erstellt und laufen, wie es für Microsoft typisch ist, auf der Hardware unter dem *.NET Micro Framework*, was sich in der Praxis als recht komplexes System darstellt. Bemerkenswert ist für Microsoft sicherlich, dass die Gadgeteer-Plattform mittlerweile als Open Source *(http://gadgeteer.codeplex.com/)* propagiert wird. Obwohl diese Plattform bereits im Jahre 2011 gestartet wurde, ist die Hardware-Auswahl und Beispielsoftware im Vergleich zu den anderen Plattfor-

men immer noch sehr begrenzt. Das Mainboard ist zudem mit ca. 100 € verhältnismäßig teuer.

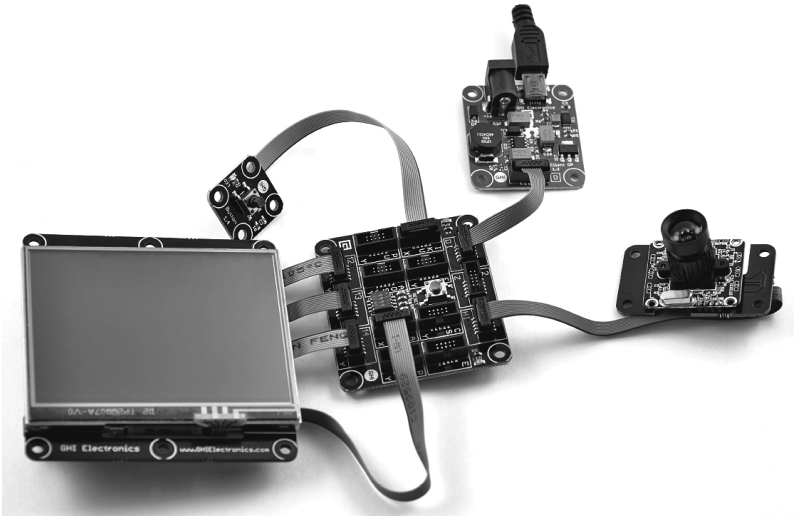

Abb. 3–21 *Der Anschluss von Peripherie erfolgt bei Gadgeteer über einzelne Flachbandkabel.*

3.5 Arduino

Arduino ist die erste bekannte Open-Source-Plattform für die Entwicklung von Mikrocontroller-Applikationen, die sich auch explizit an Einsteiger und fachfremde Benutzer richtet. Initiiert wurde Arduino im Jahre 2005 von Massimo Banzi und David Cuartieles am *Interaction Design Institute Iverea* in Italien. Wie beim Raspberry Pi ist Arduino ursprünglich ebenfalls als kostengünstige Plattform – bestehend aus quelloffener Hard- und Software – für Studenten gedacht, die hiermit eigene Projekte realisieren können und sich in einer Community hierüber austauschen, was in der Zwischenzeit zu einer geradezu unüberschaubaren Anzahl von Applikationen geführt hat. Basis einer Entwicklung ist meist ein (preisgünstiges) Arduino-Board, wovon es mittlerweile ein ganze Reihe recht unterschiedlicher Modelle gibt (siehe Tabelle 3–2).

3.5.1 Boards

Ursprünglich wurde auf einem Arduino-Board stets ein AVR-Mikrocontroller (siehe Abschnitt 2.3) der Firma Atmel mit 8 Bit und einem typischen Takt von 16 MHz eingesetzt. Beim Modell Due wird erstmalig ein ARM-basierter Con-

troller (Cortex M3) verwendet, der mit 84 MHz arbeitet und ebenfalls von Atmel stammt. Die Platinengröße entspricht der der Arduino-Mega-Typen, zu denen der Due anschlusskompatibel ist. Trotz der völlig anderen Hardware wird der Due genauso mit der IDE verwendet und programmiert wie die ursprünglichen Arduino-Typen, von denen der Uno der beliebteste ist.

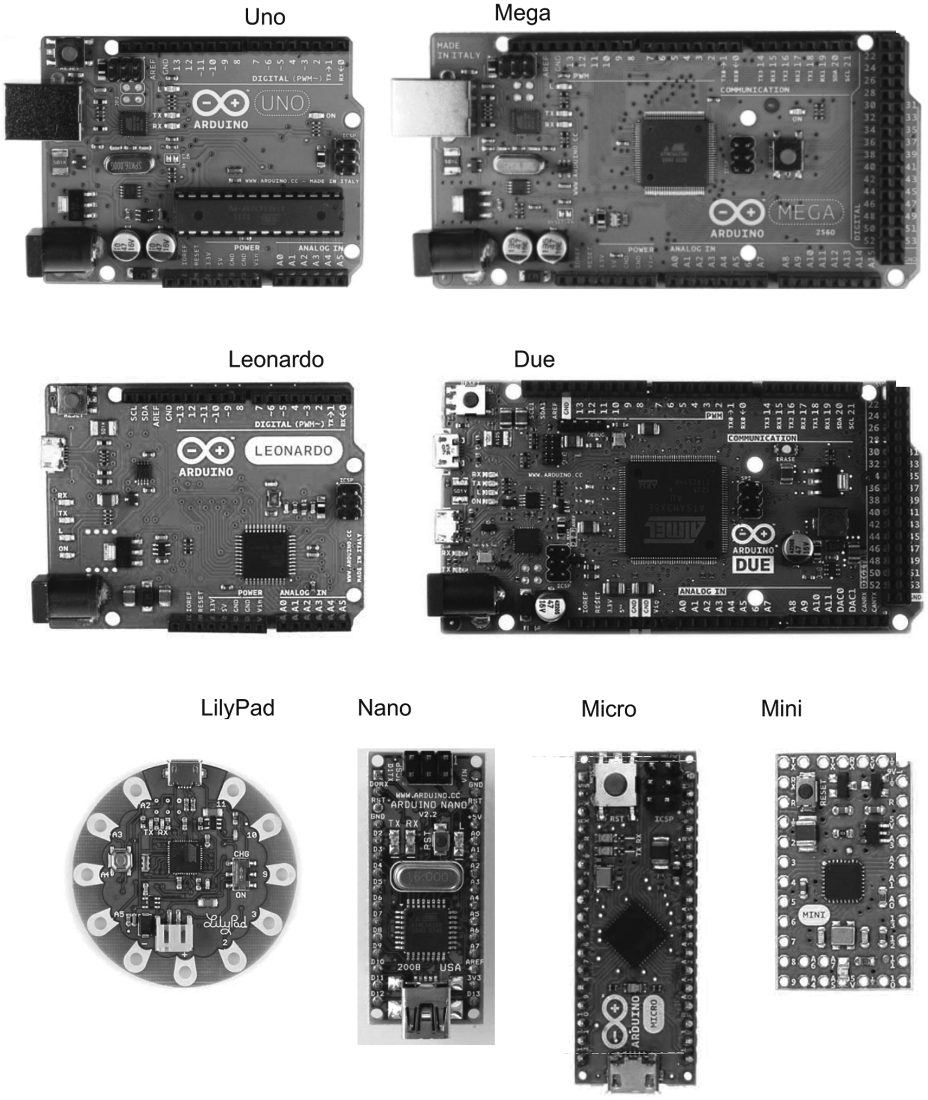

Abb. 3–22 *Arduino-Boards*