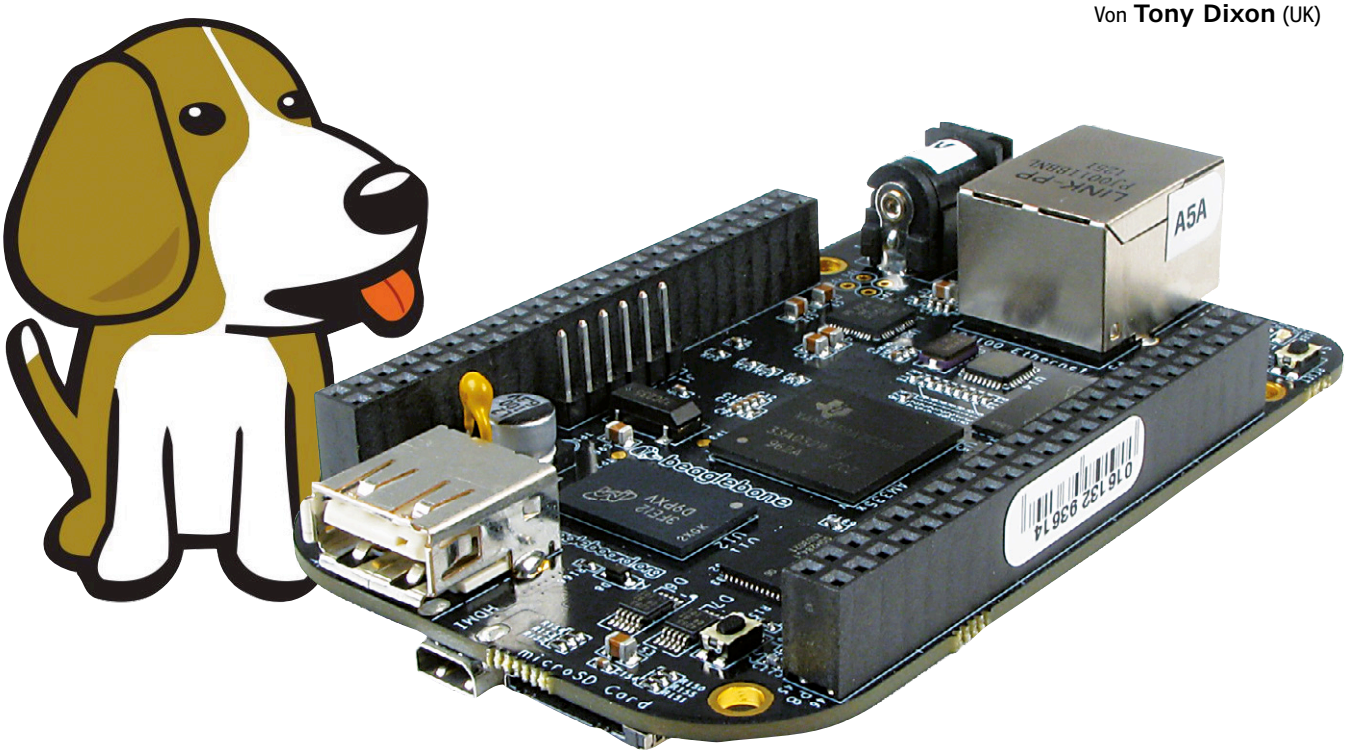
BeagleBone Black, die Serie

Teil 3: Analoge Eingänge

Von **Tony Dixon** (UK)



In der ersten Folge dieser Serie ging es um digitale I/Os. Jetzt werden die analogen Fähigkeiten des BBB (BeagleBone Black) beleuchtet.

Im Gegensatz zur Ansicht der im kalifornischen Silicon Valley ansässigen Firmen ist die Welt immer noch nicht komplett digitalisiert. Daher folgt nun eine Einführung in analoge I/Os.

Analoge Ein- und Ausgänge

Die ADCs des BBB warten mit folgenden Daten auf:

- Auflösung von 12 bit (0...4.095)
- Wandlungszeit von 125 ns
- Spannungsbereich 0...1,8 V (!!!)

Am Erweiterungs-Stecker des BBB sind sieben analoge Eingänge zugänglich. **Tabelle 1** enthält eine Übersicht der analogen Pins. Die gesamte Belegung entnimmt man **Tabelle 2**. Zusätzlich zu den analogen Signalen findet man dort Bezeichnungen wie AVCC (analoge VCC) und AGND (analoge Masse) zur Versorgung des Analogteils.

Die digitalen Leitungen sind zwar 3,3-V-kompatibel, doch an die analogen Eingänge können nur Spannungen bis zu 1,8 V angelegt werden. Man sollte also vorsichtig mit den Spannungen sein, wenn man seinen BBB nicht gleich himmeln will. Bei größeren Spannungen empfiehlt sich ein Spannungsteiler, dessen an Masse liegender Widerstand einen Wert von 1 kΩ hat.

Table 1. Analoge Eingänge am Erweiterungsstecker	
Signal (P9)	**Pin**
AIN0	39
AIN1	40
AIN2	37
AIN3	38
AIN4	35
AIN5	36
AIN6	33
AGND	34
AVCC	32

Verwendung von sysfs

Wie schon bei den GPIO-Beispielen kommen wieder die Vorteile der virtuellen Datei/Treiber-Struktur „sysfs“ von Linux zum Tragen, um die analogen Pins zu nutzen, ohne dass man hierfür eine Zeile Code schreiben müsste. Zunächst öffnet man eine Terminal-Session und beginnt mit der Aktivierung des analogen Treibers. Hierzu tippt man den folgenden Befehl ins Terminal:

```
echo cape-bone-iio > /sys/devices/
bone_capemgr.*/slots
```

Mit dem Linux-Befehl `cat` erhält man die an AIN0 gemessene Spannung in mV:

```
cat /sys/bus/iio/devices/iio\:device0/
in_voltage0_raw
```

Wenn man hingegen den rohen ADC-Wert haben will, tippt man den Befehl:

```
cat /sys/devices/ocp.2/helper.14/AIN0
```

Analoger Code

Nicht nur für einen schnellen Test eignet sich sysfs, man kann diese Operationen auch darauf aufbauend in ein C/C++-Programm verpacken.

Table 2. Komplette Pin-Belegung des BBB

Signal	P8			Signal
GND	1		2	GND
GPIO1_6	3		4	GPIO1_7
GPIO1_2	5		6	GPIO1_3
TIMER4	7		8	TIMER7
TIMER5	9		10	TIMER6
GPIO1_13	11		12	GPIO1_12
EHRPWM2B	13		14	GPIO2_26
GPIO1_15	15		16	GPIO1_14
GPIO0_27	17		18	GPIO2_1
EHRPWM2A	19		20	GPIO1_31
GPIO1_30	21		22	GPIO1_5
GPIO1_4	23		24	GPIO1_1
GPIO1_0	25		26	GPIO1_29
GPIO2_22	27		28	GPIO2_24
GPIO2_23	29		30	GPIO2_25
UART5_CTS	31		32	UART5_RTS
UART4_RTS	33		34	UART3_RTS
UART4_CTS	35		36	UART3_CTS
UART5_TXD	37		38	UART5_RXD
GPIO2_12	39		40	GPIO2_13
GPIO2_10	41		42	GPIO2_11
GPIO2_08	43		44	GPIO2_09
GPIO2_6	45		46	GPIO2_07

Signal	P9			Signal
GND	1		2	GND
3.3V	3		4	3.3V
5V	5		6	5V
5V_SYS	7		8	5V_SYS
PWR_BUTTON	9		10	SYS_RESET
UART4_RXD	11		12	GPIO1_28
GPIO4_TXD	13		14	EHRPWM1A
GPIO1_16	15		16	EHRPWM1B
I2C1_SCL	17		18	I2C1_SDA
I2C2_SCL	19		20	I2C2_SDA
UART2_TXD	21		22	UART2_RXD
GPIO1_17	23		24	UART1_TXD
GPIO3_21	25		26	UART1_RXD
GPIO3_19	27		28	SPI1_CS0
SPI1_D0	29		30	SPI1_D1
SPI1_SCLK	31		32	AVCC
AIN4	33		34	AGND
AIN6	35		36	AIN5
AIN2	37		38	AIN3
AIN0	39		40	AIN1
GPIO_20	41		42	GPIO_7
GND	43		44	GND
GND	45		46	GND

Für die folgenden Tests wird ein 5-kΩ-Poti an AVCC (Pin 32) und AGND (Pin 34) angeschlossen. Der Schleifer kommt an AIN0 (Pin 39). Nun öffnet man eine Terminal-Session und startet den Editor *nano* durch:

```
nano analogue.cpp
```

Jetzt tippt man **Listing 1** ab und sichert dieses Programm durch Eingabe von Ctrl+X, Y sowie Enter, um damit das Sichern zu bestätigen. Wer an eine digitalisierte Welt glaubt, der kann auch versuchen, das Programm „analogue.cpp“ von der Webseite zu diesem Artikel [1] zu laden. Es steckt im Archiv „130492-11.zip“. Nach dem Sichern tippt man zur Kompilierung des Programms im Terminal:

```
g++ analogue.cpp -o analogue
```

Wenn es beim Compiler-Lauf keine Fehler gegeben hat, kann man das Programm so starten:

```
./analogue
```

Jetzt sollte man sehen können, wie die Spannung am analogen Eingang einmal pro Sekunde gemessen wird. Wenn man am Poti dreht, sollten sich andere Werte ergeben. Dieser Code-Schnipsel eignet sich gut dafür, die Temperatur mit Hilfe eines TMP36 zu messen, der glücklicherweise Spannungen zwischen 0 V und 1,8 V liefert.

(130492)

Weblinks

[1] Beagle-Webseite: http://beagleboard.org

[2] www.elektor-magazine.de/post

Listing 1

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main()
{
int fd, fdstat;
char buffer[1024];

const char AIN0  [] = "/sys/bus/iio/devices/iio\:device0/in_voltage0_raw";

/* Open sysfs to Analogue input */
fd = open (AIN0, O_RDONLY);

  while (1)
  {
    /* Read Analogue input */
    fdstat = read(fd, buffer, sizeof(buffer));

    /* Print result */
    if (fdstat != -1)
    {
      buffer[fdstat] = '\0';

      /* Print string and value*/
      printf("AIN0 value = %s \n", buffer);
```

```
      lseek(fd, 0, 0);
    }

    /* Small delay */
    sleep(1);
  }

  /* Close sysfs & exit */
  close(fd);
  return 0;
}
```