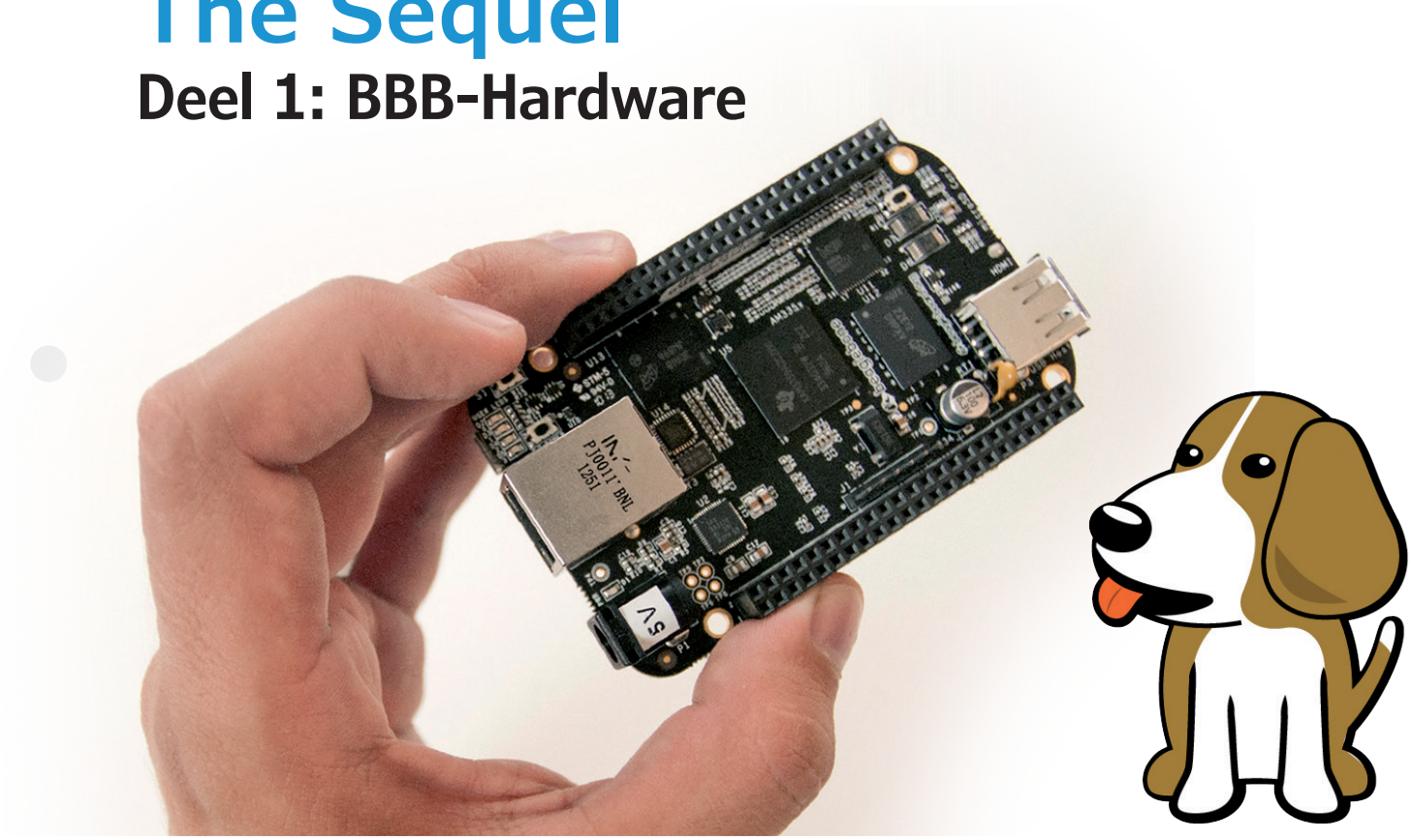
BeagleBone Black, The Sequel

Deel 1: BBB-Hardware



Tony Dixon
(Verenigd Koninkrijk)

Wie een Raspberry Pi heeft en meer I/O wil hebben en wie een Arduino Due heeft en meer verwerkingssnelheid wil, doet er goed aan om eens naar de BeagleBone Black te kijken. In deze eerste **.**Post over de BeagleBone Black (of kortweg BBB) gaan we kijken naar zijn mogelijkheden voor hardware-uitbreiding. Na een beschrijving van de hardware gaan we programmeren. We schrijven het traditionele 'Blinky' LED-knipperlicht.

Wie het introductie-artikel over de BBB van Thijs Beckers in Elektor december 2013 [1] heeft gemist, vindt in **tabel 1** een overzicht van de mogelijkheden. De BBB wordt standaard geleverd met de Ångström Linux-distributie. Het 'hondenhok' van BBB is te vinden op [2].

Voldoende hardware om kunstjes te leren

De BBB is uitgerust met flink wat hardware voor eigen projecten. Hij heeft GPIO's, analoge, PWM en seriële interfaces. De BBB heeft twee uitbreidingsconnectoren, P8

Tabel 1: Eigenschappen van de BeagleBone Black

CPU	Sitara AM3359AZC100 ARM Cortex-A8 van TI
CLOCK	1 GHz
RAM	512 MB DD3 SDRAM
VIDEO	uHDMI
GEHEUGEN	2 GB eMMC (aan boord), micro-SD-kaart
POORTEN	Ethernet 10/100Mb, USB Host, USB Client
I/O	GPIO x65, Analoog x7, PWM x8, UART x4,5, I2C x2, SPI x2
PRIJS	$45

(Expansion B) en P9 (Expansion A), beide voorzien van een hobbyist-vriendelijke 46-pens female connector met een penafstand van 0,1". In **tabel 2** is de pinbezetting van de uitbreidingsconnectoren na het inschakelen te zien. Het is mogelijk andere signalen toe te wijzen aan de pennen: zie het *BBB System Reference Manual* voor meer informatie.
De I/O-lijnen van de BBB zijn alleen geschikt voor 3,3-V-signalen, dus sluit geen 5-V-schakelingen aan, anders gaat de BBB rechtstreeks naar de hondenhemel.

Software-kennel

Omdat de BBB een Linux-computer is, kunnen we kiezen uit een groot aantal programmeertalen. Naast oude bekenden zoals C/C++ en Python kent de BBB ook een eigen taal: BoneScript. BoneScript is ontwikkeld voor de oudere broers van de BBB: BeagleBoard, BeagleBoard-XM en de oorspronkelijke BeagleBone.
BoneScript is een bibliotheek gebaseerd op Node.js, die veel bekende Arduino-achtige functies kent om met de BBB-hardware te praten. BoneScript is gebaseerd op Javascript en heeft, net als Arduino, een IDE die luistert naar de naam Cloud9 voor het ontwikkelen van programma's.

Blinky _._._._

Voor ons eerste BBB-programma sluiten we aan bij de traditie voor embedded systemen: we gaan een LED laten knipperen. We hadden BoneScript kunnen gebruiken voor dit voorbeeld, maar we kiezen voor het bekende C/C++. We sluiten voor het Blinky-programma een LED via een weerstand van 680 ohm aan op GPIO1_6 op connector P8.03.
Op de BBB worden GPIO's aangestuurd in blokken van 32 die zijn genummerd vanaf 0. Om het GPIO-nummer te berekenen vermenigvuldigen we het bloknummer met 32 en tellen daar het lijnnummer bij op. Voor ons voorbeeld met GPIO1_6 is dat dus 1*32 + 6 = 38.

Tabel 2: BeagleBone Black pinbezetting van de uitbreidingsconnectoren P8, P9.

Signaal	P8			Signaal
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

Signaal	P9			Signaal
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

Linux behandelt bijna alles als files, ook hardware-poorten zoals UART en USB. Daardoor kan een programmeur de GPIO ook via de Linux-kernel aansturen alsof het een bestand is. We kunnen de volgende file-descriptors gebruiken om de GPIO aan te sturen:

```
/sys/class/gpio/export
/sys/class/gpio/gpio38/direction
/sys/class/gpio/gpio38/value
/sys/class/gpio/unexport
```

Start een terminal-sessie en start dan de **nano**-editor. Type in de terminal:

```
nano blinky.cpp
```

Type of kopieer de code getoond in **listing 1** aan het eind van dit artikel. Als de code is ingevoerd, sla het programma dan op door Ctrl+X, Y en ENTER in te voeren.

Als het programma is opgeslagen, kunnen we het vanuit de terminal compileren met het commando:

```
g++ blinky.cpp -o blinky
```

Als het programma zonder fouten is gecompileerd, kunnen we het uitvoeren met het commando:

```
./blinky
```

Nu moet onze LED rustig aan en uit gaan knipperen in een tempo van eenmaal per seconde. U ziet het, “Pas op voor de hond” is op de BeagleBone Black nauwelijks van toepassing.

(130472)

Weblinks

[1] BeagleBone Black, Elektor december 2013, www.elektor-magazine.nl/130279

[2] Beagle-website: http://beagleboard.org

Listing 1: blinky.cpp

```
#include <stdio.h>
#include <unistd.h>

using namespace std;

int main() {
FILE *export_file = NULL;
FILE *IO_dir = NULL;
char str_low[] = “low”;
char str_high[] = “high”;
char str_port[] = “38”;

// Open Port
export_file = fopen (“/sys/class/gpio/export”, “w”);
fwrite (str_port, 1, sizeof(str_port), export_file);
fclose (export_file);

  while (1) {
          IO_dir = fopen (“/sys/class/gpio/gpio38/direction”, “w”);
          fwrite (str_high, 1, sizeof(str_high), IO_dir);   // pin = HIGH
          fclose (IO_dir);
          sleep (1);

          IO_dir = fopen (“/sys/class/gpio/gpio38/direction”, “w”);
          fwrite (str_low, 1, sizeof(str_low), IO_dir);   //pin = LOW
          fclose (IO_dir);
          sleep (1)
  }

}
```