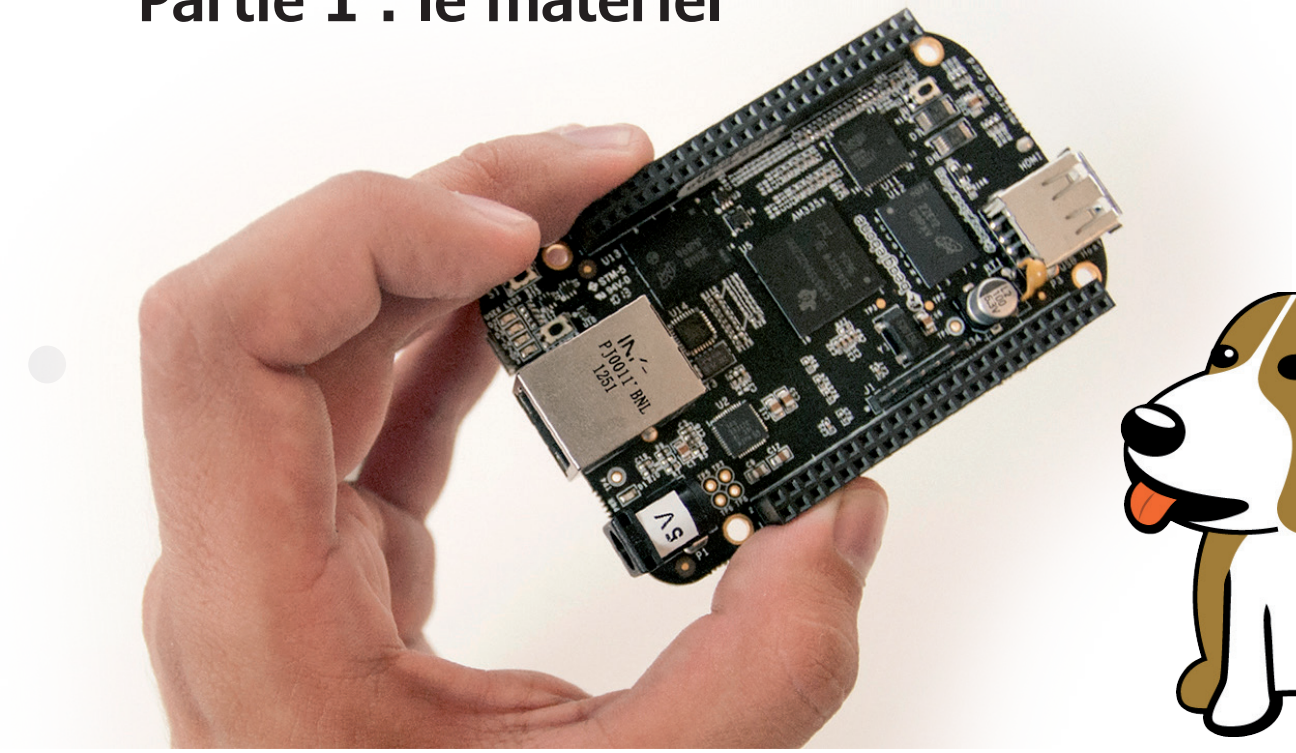
BeagleBone Black

Partie 1 : le matériel



Tony Dixon (Royaume-Uni)

Si vous possédez un Raspberry Pi et avez besoin de plus d'E/S, ou si la vitesse de traitement de votre Arduino Due ne vous suffit pas, la plate-forme BeagleBone Black vous conviendra sans doute. Dans ce premier .Post consacré au BeagleBone Black (BBB), nous passerons en revue ses capacités d'extension matérielle, puis nous écrirons un premier programme qui, devinez, fera clignoter une LED.

Nous ne présenterons pas ici tous les atouts matériels du BBB, Thijs Beckers l'a déjà fait dans le numéro de novembre d'Elektor (p. 46) [1]. Rappelons juste que le BBB est livré par défaut avec une distribution Linux Angström pré-installée, et que vous pouvez le dénicher sur le site [2].

Toutou ce qu'il faut

Le BBB saura être le précieux compagnon de vos projets matériels. Ses papattes sont pleines de GPIO, d'analogique, de MLI et d'interfaces série. Il est équipé de deux connecteurs d'extension P8 et P9 (également appelés *Expansion A* et *Expansion B*) de 48 contacts femelles au pas de 2,54 mm.

Le **tableau 2** montre leur brochage disponible après mise sous tension du BBB. Il est possible d'affecter d'autres signaux aux bro-

Tableau 1. Caractéristiques du BeagleBone Black

CPU	Sitara AM3359AZC100 ARM Cortex-A8 de TI
HORLOGE	1 GHz
RAM	512 Mo DD3 SDRAM
VIDÉO	uHDMI
STOCKAGE	2 Go de flash eMMC (sur la carte), carte micro SD
PORTS	Ethernet 10/100 Mbit/s, hôte USB, client USB
E/S	GPIO x65, analogique x7, PWM x8, UART x4,5, I²C x2, SPI x2
PRIX	39 € HT

ches ; référez-vous au manuel de référence pour découvrir les options de multiplexage.

Les E/S du BBB sont toutes des signaux de niveau 3,3 V, donc évitez de les relier à des circuits de 5 V si vous ne voulez pas transformer votre BBB en vieil os.

La moelle logicielle

Puisque le beagle est aussi un pingouin, l'éventail des langages de programmation à notre disposition est large. On peut bien sûr utiliser C/C++ ou Python, mais aussi BoneScript, un langage apparu sur les cartes parentes du BBB, à savoir BeagleBoard, BeagleBoard-XM, et BeagleBone, la carte originale.

BoneScript est une bibliothèque basée sur Node.js. Un bon nombre de ses appels de fonction ressemblent à ceux d'Arduino et permettent d'interagir avec le matériel du BBB. BoneScript est enracinée dans JavaScript et a son propre EDI, appelé Cloud9.

Mon chien cligne de l'œil

Respectons la tradition propre à la programmation des systèmes embarqués : inaugurons notre premier programme avec le clignotement d'une LED. Nous aurions pu écrire le code en BoneScript, mais nous avons préféré le C/C++, sans doute plus familier à beaucoup. La LED est reliée via une résistance-talon de 680 ohms à la broche GPIO1_6 du connecteur P8.03.

Les ports GPIO du BBB sont contrôlés par blocs de 32, l'indice du premier bloc étant 0. Pour obtenir le numéro d'un port GPIO, il faut multiplier son numéro de bloc par 32 et ajouter le numéro de signal. Le numéro de GPIO1_6 est donc : 1*32 + 6 = 38.

Dit rapidement, Linux traite pratiquement tout comme un fichier, y compris les ports matériels tels qu'UART et USB. Le programmeur peut ainsi accéder à un port GPIO via le noyau Linux comme s'il s'agissait d'un descripteur

Tableau 2. Brochage des connecteurs d'extension P8 et P9 du 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

de fichier. Nous utilisons ici les descripteurs suivants :

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

Ouvrez un terminal, et appelez l'éditeur **nano** à l'aide de la commande :

```
nano blinky.cpp
```

Recopiez ou collez le code du **listage 1**, et enregistrez le fichier avec Ctrl+X, puis Y et Entrée pour confirmer la sauvegarde.

Pour compiler ce programme C/C++ depuis le terminal, entrez :

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

Si la compilation s'est déroulée sans erreur, exécutez le programme avec :

```
./blinky
```

La LED clignote paisiblement, une fois par seconde. Finalement, malgré son nom évocateur de pirate borgne, il n'y a aucune raison d'associer l'avertissement « Attention au chien » à ce BeagleBone Black, vous ne croyez-pas ?

(130472 – version française : Hervé Moreau)

Liens

[1] « et voici BeagleBone Black », Elektor, novembre 2013 ; www.elektor-magazine.fr/130279

[2] Le site de Beagle : http://beagleboard.org

Listage 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)
  }

}
```