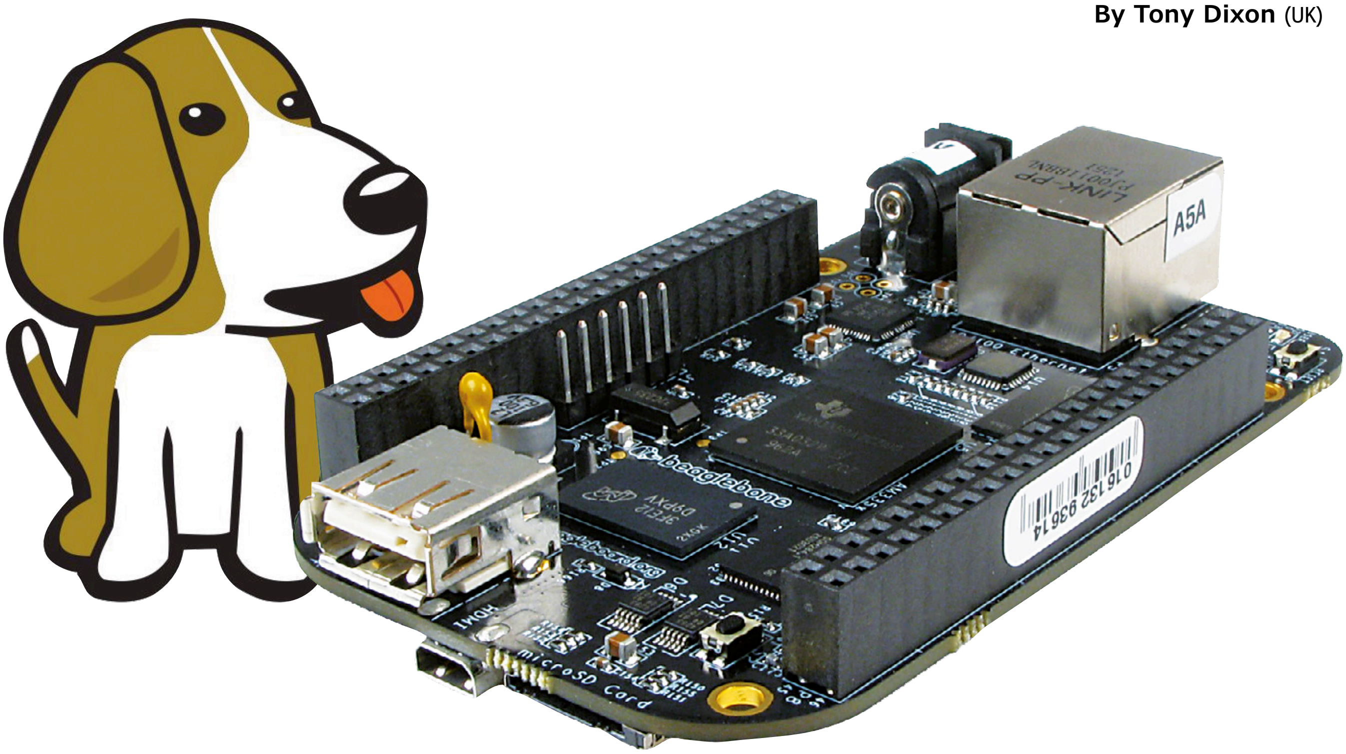
BeagleBone Black, The Sequel (3)

Part 3: BBB Analog Inputs

By Tony Dixon (UK)



In our first dot-Post on the BeagleBone Black (BBB) we looked at digital I/O. In this installment we'll deal with the BBB's analog capabilities. Let's find our [USB] lead and take the Dog for a walk.

Contrary to what some people in Silicon Valley and others always seen near Ethernet outlets want to make you believe, the world is not entirely digital.

Introducing the BBB Analog I/O

The BBB ADC has the following properties:

- 12-bit resolution (0 to 4095)
- 125-ns sample time
- 0 V to 1.8 V range (!!!!)

There are 7 analog inputs available on the BBB expansion connectors. See **Table 1 f**or a quick summary of analog pins and for the BBB pin-out details in full, **Table 2**.
In addition to the analogue signals there are also separate AVCC (Analogue VCC) and AGND (Analog Ground) power supplies pins.
Whilst the BBB GPIO are 3.3-V compatible, the analog pins are only rated at 1.8 V. So be careful what voltage signals you connect to them unless we want to send your BBB to the great kennel in the sky. If you plan to measure anything greater than 1.8 V use a voltage divider with a lower leg resistor value of 1 k-ohms.

Table 1. Analog port pinout	
Signals (P9)	**Pin**
AIN0	39
AIN1	40
AIN2	37
AIN3	38
AIN4	35
AIN5	36
AIN6	33
AGND	34
AVCC	32

Using sysfs

Like the earlier GPIO examples we again have the advantage of being able to use Linux's 'sysfs' virtual file/driver structure to interact with the analog pins without resorting to writing a single line of code.

Let's open a terminal session and start by enabling the analog driver. Type the following command in the terminal session:

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

Using the Linux command `cat` we can report (or measure) the voltage in millivolts (mV) at AIN0 by typing:

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

If we want to see the ADC count instead, we can use the following command:

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

~~Feeding~~ Coding Time

Whilst using sysfs is great for a quick test we can build on this and wrap these operations into a C/C++ program.

For our test we'll use a 5-kΩ potentiometer connected across AVCC (pin 32) and AGND

Table 2. BeagleBone Black Expansion Pinouts; P8, P9.

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

(pin 34) with the wiper connected to AIN0 (pin 39). Open a terminal session and start the *nano* editor with:

```
nano analogue.cpp
```

Type the program from **Listing 1** appended to the article. Once finished, save the program by pressing Ctrl+X, Y and Enter to confirm saving the program. Silicon Valley people only: download the program 'analogue.cpp' from our website [1], it's in archive file 130492-11.zip.
Once saved, in our terminal we can compile the C/C++ program by typing:

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

Once compiled if we've had no compilation errors we can run our program by typing:

```
./analogue
```

We should see the analog pin being measured once a second. Turn the pot and observe the screen print out.
We could easy use this code snippet to measure temperature by using a TMP36 which by some good fortune has an output 0 V to 1.8 V.

(130492)

Web Links

[1] Beagle Website: http://beagleboard.org
[2] www.elektor-magazine.com/130492

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;
}
```