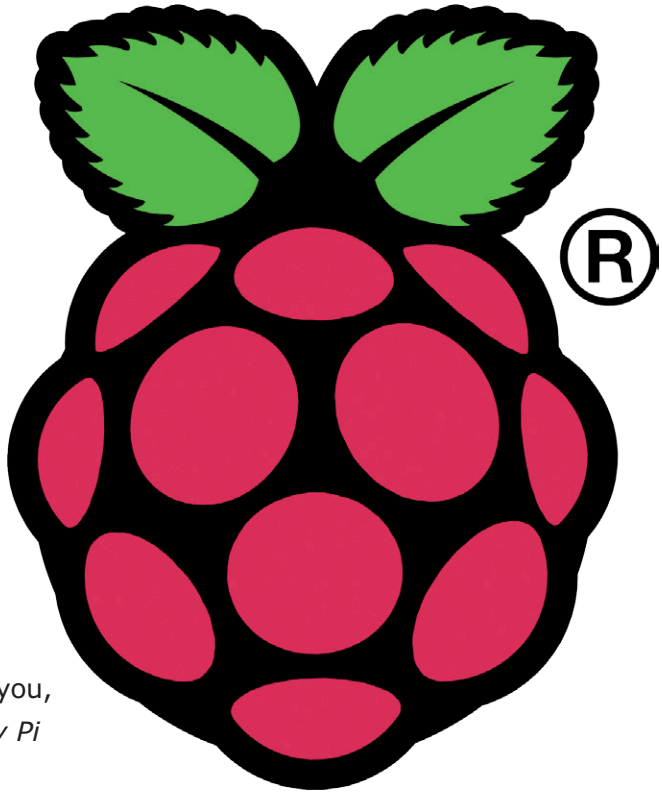
Raspberry Pi Recipes Part #2

No harm in having a few extra kitchen appliances ready



By **Tony Dixon** (UK)

In the first **.**POST on Rpi e-cookery we showed how to install Raspbian and how to set up our Raspberry Pi to get us started. As promised last time, in this part we'll be looking at the Expansion Header of the Raspberry Pi and how to program the GPIO pins we'll find there. If you've read Elektor's March 2013 edition some of this may be familiar to you, as we discussed the Expansion Port in the *Raspberry Pi Prototyping Board* article [1].

RPi Expansion Header: double the taste

For hardware hackers everywhere, the Raspberry Pi Expansion Header has to be the most exciting thing on the Pi, after the Pi's stupendously low cost of course.

You'll find expansion header in the corner near the composite video connector. It's a hobbyist friendly double row, 13-way 0.1" (2.54mm) pinheader connector so it makes interfacing to it easy.

The 26-pin Expansion Header provides three categories of signals:

- Power: +5 V DC and 3.3 V DC* as well as 0 V
  (Please note that the 3.3 V rail can only provide about 50 mA of current)
- Input/Output: General Purpose Input/Output (GPIO) signals
- Communications Interfaces: Serial UART, SPI and I²C

There are 17 general purpose input / output (GPIO) signals on the expansion header. Most of these can have an alternative function. These alternative functions provide a UART, SPI and I²C interfaces — see **Table 1**.

Each GPIO pad can source between 2 and 16 mA depending on its drive strength configuration. The drive strength is set in a configuration register and by default after reset the source current is set to 8 mA.

Revision 2 of the Raspberry Pi saw the introduction of a second, smaller expansion header, P5 — see **Table 2**. This adds another four GPIO signals but more importantly for audio aficionados it allows access to the PCM audio interface of the Broadcom 2835 chip.

Besides adding a new expansion interface, revision 2 Pis saw the signals for P1 Expansion revised slightly, with the P1's I2C0 interface replaced by the I2C1 interface. A small but important thing to remember if you are planning to interface I²C devices to you Pi.

Table 1. Expansion Header Pinout

Pin	Function	Alternative	RPi.GPIO
P1-02	5.0 V	-	-
P1-04	5.0 V	-	-
P1-06	GND	-	-
P1-08	GPIO14	UART0_TXD	RPi.GPIO8
P1-10	GPIO15	UART0_RXD	RPi.GPIO10
P1-12	GPIO18	PWM0	RPi.GPIO12
P1-14	GND	-	-
P1-16	GPIO23		RPi.GPIO16
P1-18	GPIO24		RPi.GPIO18
P1-20	GND	-	-
P1-22	GPIO25		RPi.GPIO22
P1-24	GPIO8	SPI0_CE0_N	RPi.GPIO24
P1-26	GPIO7	SPI0_CE1_N	RPi.GPIO26

Pin	Board Revision 1		Board Revision 2	
	Function	Alternative	Function	Alternative
P1-01	3.3 V	-	3.3 V	-
P1-03	GPIO0	I2C0_SDA	GPIO2	I2C1_SDA
P1-05	GPIO1	I2C0_SCL	GPIO3	I2C1_SCL
P1-07	GPIO4	GPCLK0	GPIO4	GPCLK0
P1-09	GND	-	GND	-
P1-11	GPIO17	RTS0	GPIO17	RTS0
P1-13	GPIO21		GPIO27	
P1-15	GPIO22		GPIO22	
P1-17	3.3 V	-	3.3 V	-
P1-19	GPIO10	SPI0_MOSI	GPIO10	SPI0_MOSI
P1-21	GPIO9	SPI0_MISO	GPIO9	SPI0_MISO
P1-23	GPIO11	SPI0_SCLK	GPIO11	SPI0_SCLK
P1-25	GND	-	GND	-

Note: I2C0_SDA and I2C0_SCL (GPIO0 & GPIO1), and I2C1_SDA and I2C1_SCL (GPIO2 & GPIO3) have 1.8 kΩ (1k8) pull-up resistors to 3.3 V (3V3).

Table 2. P5 Header Pinouts

Pin	Function	Alternative
P5-01	5.0 V	
P5-02	3.3 V	
P5-03	GPIO28	PCM_CLK
P5-04	GPIO29	PCM_FS
P5-05	GPIO30	PCM_DIN
P5-06	GPIO31	PCM_DOUT
P5-07	GND	
P5-08	GND	

Figure 1. LXTerminal

Installing Python's GPIO Library

We are going to program our examples in Python. Fortunately Python is installed as standard in the Raspbian distribution but to access the Pi's GPIO we'll need to install a suitable hardware I/O library. There are a few libraries we could use but we'll be using the Python RPi.GPIO library to give use access to the GPIO pins.

If you've not already downloaded the Python development tools or Python GPIO library then using an LX Terminal (see **Figure 1**) on your Pi, we'll first download the Python development tools by typing:

```
sudo apt-get install python-dev
```

Next, we'll install the Python GPIO Library package [1] by typing the following:

```
wget http://pypi.python.org/
packages/source/R/RPi.GPIO/RPi.GPIO-
0.5.0a.tar.gz
```

Once downloaded, we'll need to extract the files. Type:

```
tar -zxf RPi.GPIO-0.5.0a.tar.gz
```

After this, a new a directory will be created with the Python files in. Now type:

```
cd RPi.GPIO-0.5.0a
```

Now we'll install the package by typing:

```
sudo python setup.py install
```

Once that's done we should have the Python RPi.GPIO library installed.

Example program: blinky.py

With RPi.GPIO installed we're now going to write a small test program to flash an LED. **Figure 2** shows our setup. We've used a small breadboard (MiniPiio ProtoBoard [2]) with our Pi, and we've wired a LED and a 680 Ω resistor from GPIO17 (pin 11) to 0V. After the components are fitted we'll double click IDLE icon on your Pi's desktop to start the Python Shell and IDE (**Figure 3**).

Select the File option from the menu and create a new program. This will start the IDE editor.

In the IDLE editor (**Figure 4)** type the program as shown in **Listing 1**.

After typing the program, make sure you save it, then switch to an LX Terminal and type the following command to make your program an executable:

```
chmod +x blinky.py
```

Once done, you can run your program by typing the following command:

```
sudo ./blinky.py
```

Tip: if you start IDLE from an LXTerminal session with **sudo** preceding the program name e.g. **sudo idle**, you will have the correct permission to run your RPi.GPIO program from within IDLE.

RPi.GPIO pin numbering

RPi.GPIO has two ways of numbering the GPIO signal pins on the Raspberry Pi. So your RPi.GPIO program must always specify a GPIO number system, either GPIO.setmode(GPIO.BOARD) or GPIO.setmode(GPIO.BCM).

The first is using the GPIO.setmode(GPIO.BOARD) numbering system, which refers to the pin numbers on the P1 expansion header of the Raspberry Pi board. The advantage of

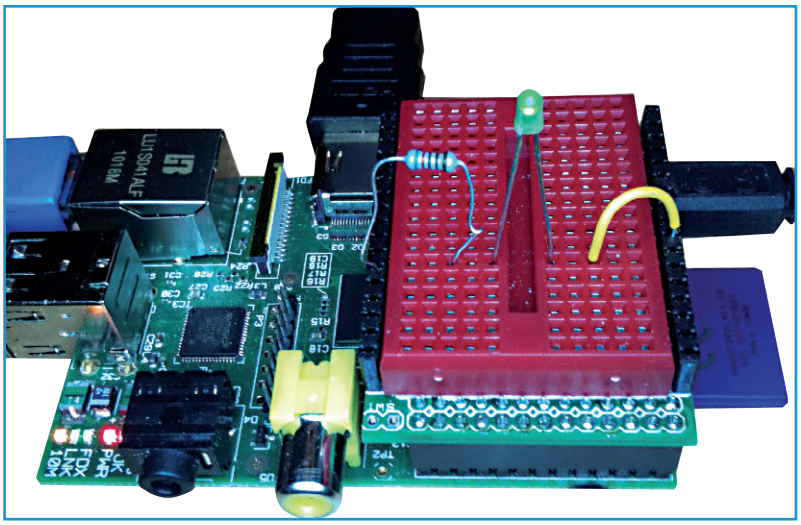


Figure 2.
Pi and breadboard.

Listing: blinky.py

```
# !/usr/bin/python
import time
import RPi.GPIO as GPIO

# Configure Pi's GPIO pins
GPIO.setmode (BCM)
GPIO.setup (17,GPIO.OUT)

# Program loop
while True :
  GPIO.output (17,True)
  time.sleep (1)
  GPIO.output (17,True)
  time.sleep (1)
```

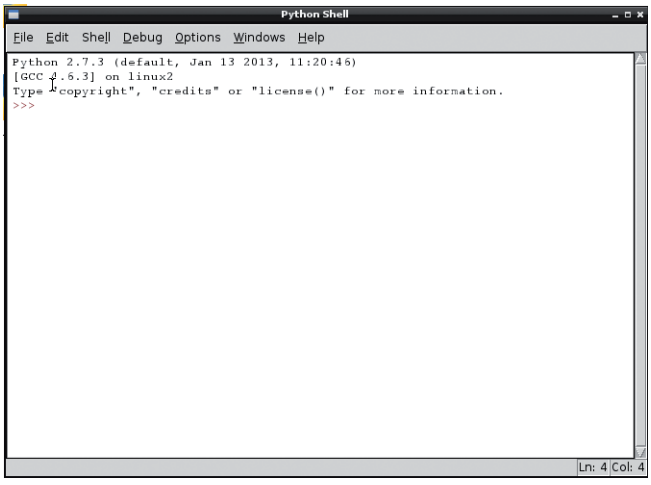


Figure 3.
IDLE Python Shell.

this is you will not need to change your code if any future changes are made to the signals of the Raspberry Pi Expansion Header.
The second method is the GPIO.setmode(GPIO.BCM) numbering, which references the Broadcom 2835 signal names directly.
This can be a little confusing at times, so **Table 3** shows how P1 pin numbers, GPIO signal names and GPIO.setmode all fit together.

(130110)

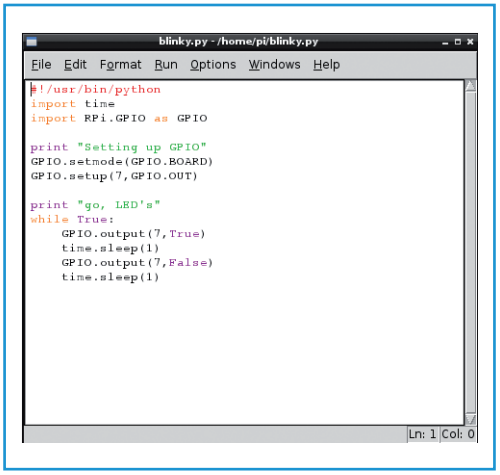


Figure 4.
IDLE Editor.

Internet Links

[1] www.elektor.com/120483

[2] https://pypi.python.org/pypi/RPi.GPIO

[3] www.dtronixs.com

Table 3. GPIO.setmode(GPIO.BCM) and GPIO.setmode(GPIO.BOARD)

Pin	Function	GPIO.setmode	
		GPIO.BCM	GPIO.BOARD
P1-01	3.3 V	-	-
P1-02	5.0 V	-	-
P1-03	GPIO0/2*	0/2	3
P1-04	5.0V	-	-
P1-05	GPIO1/3*	1/3	5
P1-06	GND	-	-
P1-07	GPIO4	4	7
P1-08	GPIO14	14	8
P1-09	GND	-	-
P1-10	GPIO15	15	10
P1-11	GPIO17	17	11
P1-12	GPIO18	18	12
P1-13	GPIO21/27*	21	13
P1-14	GND	-	-
P1-15	GPIO22	22	15
P1-16	GPIO23	23	16
P1-17	3.3 V	-	-
P1-18	GPIO24	24	18
P1-19	GPIO10	10	19
P1-20	GND	-	-
P1-21	GPIO9	9	21
P1-22	GPIO25	25	22
P1-23	GPIO11	11	23
P1-24	GPIO8	8	24
P1-25	0 V	-	-
P1-26	GPIO7	7	26

Note: * Revision 2 changes.