Wiring Pi

GPIO Interface library for the Raspberry Pi

SPI Library

WiringPi includes a library which can make it easier to use the Raspberry Pi's on-board SPI interface.

Before you can use SPI interface, you may need to use the **gpio** utility to load the SPI drivers into the kernel:

```
gpio load spi
```

If you need a buffer size of greater than 4KB, then you can specify the size (in KB) on the command line:

```
gpio load spi 100
```

will allocate a 100KB buffer. (You should rarely need this though, the default is more than enough for most applications).

To use the SPI library, you need to:

```
#include <wiringPiSPI.h>
```

in your program. Programs need to be linked with **-lwiringPi** as usual.

Functions available:

- **int wiringPiSPISetup (int channel, int speed) ;**

This is the way to initialise a channel (The Pi has 2 channels; 0 and 1). The speed parameter is an integer in the range 500,000 through 32,000,000 and represents the SPI clock speed in Hz.

The returned value is the Linux file-descriptor for the device, or -1 on error. If an error has happened, you may use the standard *errno* global variable to see why.

- **int wiringPiSPIDataRW (int channel, unsigned char *data, int len) ;**

This performs a simultaneous write/read transaction over the selected SPI bus. Data that was in your buffer is overwritten by data returned from the SPI bus.

That’s all there is in the helper library. It is possible to do simple read and writes over the SPI bus using the standard read() and write() system calls though – write() may be better to use for sending data to chains of shift registers, or those LED strings where you send RGB triplets of data. Devices such as A/D and D/A converters usually need to perform a concurrent write/read transaction to work.