Raspberry Pi Recipes Part #6

A Distinguished Chef enters the kitchen

By **Tony Dixon** (UK)

So far in our Elektor.POST projects we have looked at mainly digital signals such as GPIO, Serial UART, SPI and I²C of the Raspberry Pi's expansion header. In this installment we'll be adding some analog functionality our Raspberry Pi's by adding an Analog-to-Digital (ADC) interfaces via the SPI Bus.

A lack of (analog) culinary tools ☹

The Expansion Header of the Raspberry Pi does not have any analog interfaces, not a single one. This is a shame, when other platforms such as the Arduino and BeagleBone Black all sport several of them.

But do not despair, we can use either one (or both) of the SPI or I²C serial interfaces to connect a serial ADC. So let's start using the SPI interface. We'll be adding a 4-channel MCP3004 ADC chip with 10-bit resolution.

SPI Interface

We previously looked at the Serial Peripheral Interface (SPI) Interface in the Elektor.POST project #9 but we can quickly recap on the hardware interface.

Table 1 details the Expansion Header signals and the SPI interface can be found on pin 19 (MOSI), pin 21 (MISO), pin 23 (SCK) with the two SPI Chip Enables found on pins 24 (CE0) and 26 (CE1).

Check out the text box *"Installing Python's SPI Library"* for details how to install the Python SPI library and how to configure your Pi to access the SPI interfaces.

ADC Hardware

In this, our Raspberry Pi first analog project we'll be using a single MCP3004 from Microchip [1]. The MCP3004 is a 4-channel, 10-bit ADC.

Figure 1 shows a simple MCP3004 circuit. The chip is connected to the RPi's SPI interface and we have the option of using either one off the two SPI Chip Enable sig-

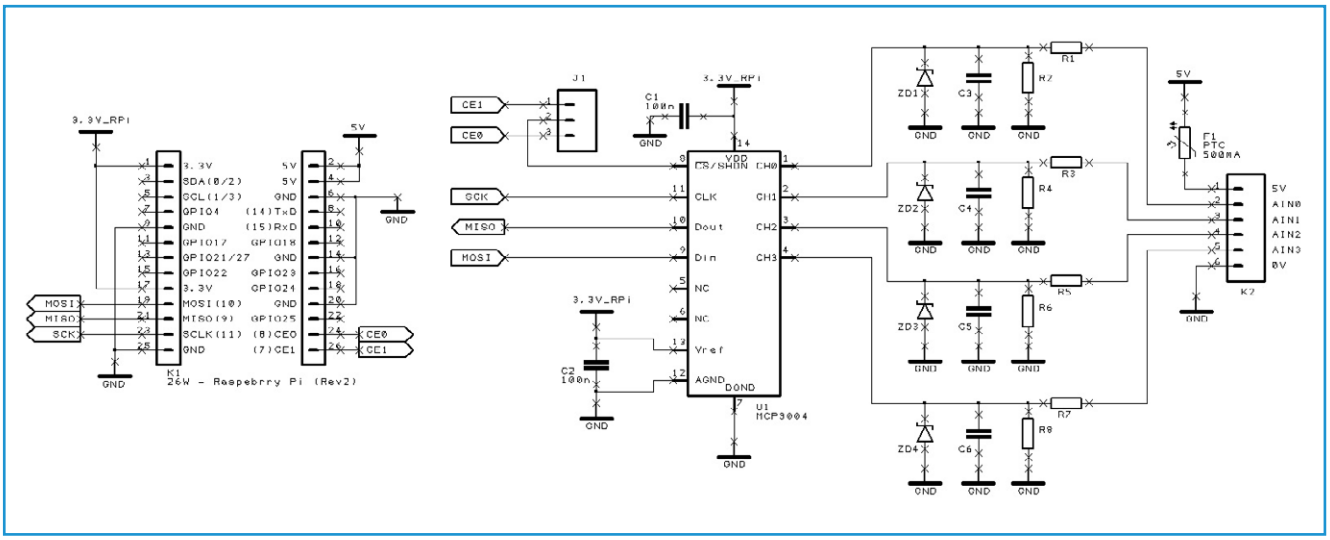

Figure 1. Schematic for Raspberry Pi MCP3004 ADC Interface.

nals (CE0 or CE1) via a jumper. The circuit is easy enough to assemble in a standard breadboard.

The MCP3004 is powered from 3.3 volts, and we should take precautions with what voltages we connect to it. The signal condition circuit for each of the ADC channels is the same. If we look at Channel 1 of the ADC we can see a voltage divider circuit performed by resistors R1 and R2. Using the voltage divider calculation:

$$V_{out} = \frac{R2}{R1 + R2} \times V_{in}$$

- To measure 0V to 5 V, R1 = 10 kΩ and R2 = 10 kΩ, which would give us an ADC reading of 0 V – 2.5 V.
- To measure 0 V to 10 V, R1 = 10 kΩ and R2 = 22 kΩ, which would give us an ADC reading of 0V to 3.125 V.
- To measure 0 V to 3.3 V, R1 = 1 kΩ and R2 is not fitted. This would give us an ADC reading of 0 V to 3.3 V.

We can repeat the resistors choices for the other ADC channels #2, #3 and #4 as needed.
If we need to add a little filtering to our analog signal to perhaps get rid of unwanted noise we use a value of 1 nF to 10 nF for capacitors C3–C6.

If we want to add additional overvoltage protection to our circuit we can populate

Table 1. Expansion Header Pin Out

Pin Name	Pin Function	Alternative	RPi.GPIO
P1-02	5.0V	-	-
P1-04	5.0V	-	-
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	
Name	Pin Function	Alternative	Pin Function	Alternative
P1-01	3.3V	-	3.3V	-
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	3.3V	-	3.3V	-
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 pull-up resistors to 3.3 V.

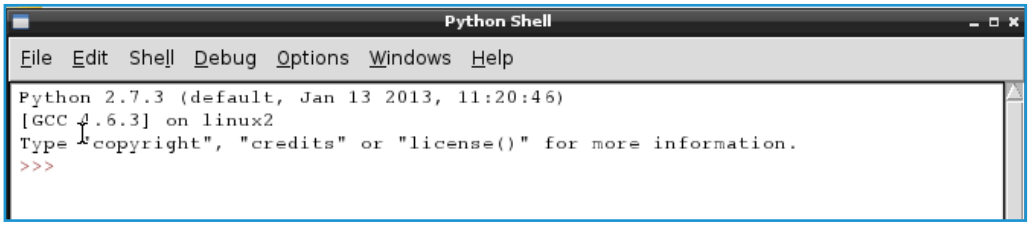

Figure 2.
IDLE Python Shell.

ZD1 to ZD4 with 3.3-V zener diodes.

Example Program – mcp3004.py
With our circuit built and spidev installed (see textbox "Installing Python's SPI Library") we're now going to write a small test program to measure and display a voltage on ADC Channel 1.
Double-click the IDLE icon on your Pi's desktop to start the Python Shell and IDE

Installing Python's SPI Library

We already installed the Python SPI library [2] back in Elektor.POST project #9, but in case you missed it here are the instructions again. Let's start an LXTerminal session (Figure 4) and type the following commands:

```
sudo apt-get install git-core

cd ~
git clone git://github.com/doceme/py-spidev
cd py-spidev/
sudo python setup.py install
```

or alternatively you can use the "Pip", the Python Package Installer

```
sudo apt-get install git-core python-dev
sudo apt-get install python-pip
sudo pip install spidev
```

The hardware SPI is disabled by default, so we have to change this by editing the blacklist file:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Find the text line with blacklist spi-bcm2708, insert a # (hash) at the start of the line to comment out the statement and then save the file. Once saved, let's do a quick reboot by typing

```
sudo reboot
```

Once we've rebooted, we can start a new LXTerminal session and type...

```
ls /dev/spi*
```

...to check that we have two SPI devices list (one for each SPI Chip Select signal) and we should have:

```
/dev/spidev0.0
/dev/spidev0.1
```

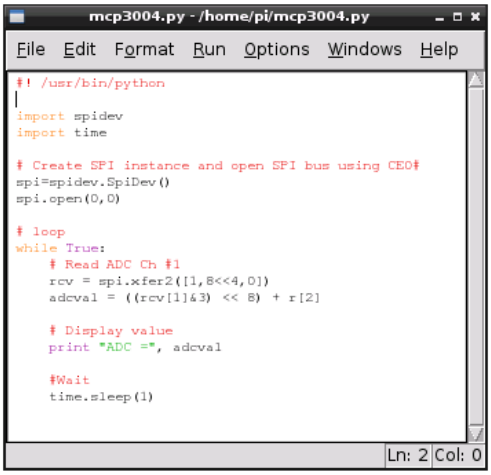

Figure 3. IDLE Editor.

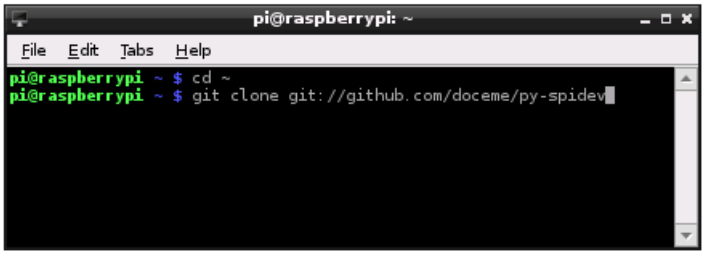

Figure 4. LXTerminal.

(**Figure 2**).
Select File option from the menu and create a new program. This will start the IDLE editor. In the IDLE editor (**Figure 3**), type the program as shown in **Listing 1**.
Once you've typed the program, make sure you save it, then switch to a LXTerminal and type the following command to make your program an executable:

```
chmod +x mcp3004.py
```

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

```
sudo ./mcp3004.py
```

(130260)

Internet Links

[1] http://goo.gl/eMSOQ

[2] https://github.com/doceme/py-spidev

Listing 1

```
#! /usr/bin/python

import spidev
import time

# Create SPI instance and open SPI bus using CE0
spi = spidev.SpiDev()
spi.open(0,0)

# Loop
while True:

    # Read ADC Ch #1
    rcv = spi.xfer2 ([1,8<<4, 0])
    adcval = ((rcv [1]&3) << 8) + rcv[2]

    # Display value
    Print "ADC = ", adcval

    # Wait
    time.sleep(1)
```

A Quick Summary of spidev commands	
spi.open (0,0)	opens SPI bus 0 using CE0.
spi.open (0,1)	opens SPI bus 0 using CE1.
spi.close ()	disconnects the object from the interface.
spi.writebytes ([array of bytes])	writes an array of bytes to SPI device.
spi.readbytes (len)	reads *len* bytes from SPI device.
spi.xfer2 ([array of bytes])	transmits an array of bytes keeping the CEx asserted the whole time.
spi.xfer ([array of bytes])	transmits an array of bytes de-asserting and re-asserting the CEx with every byte transmitted.