

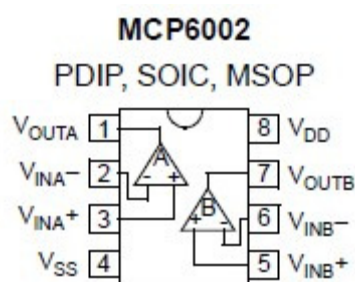
Project 13 - Using the Raspberry Pi to measure battery voltage

Outline

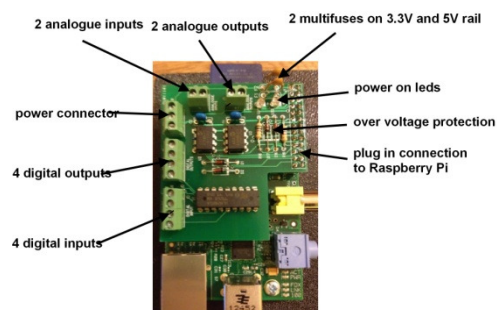
This application note shows the hardware and Python code required to measure the voltage of a 9V PP3 battery using the Raspberry Pi. Any of the Raspberry Pi models can be used including B, B+ and the Raspberry Pi 2.

Hardware

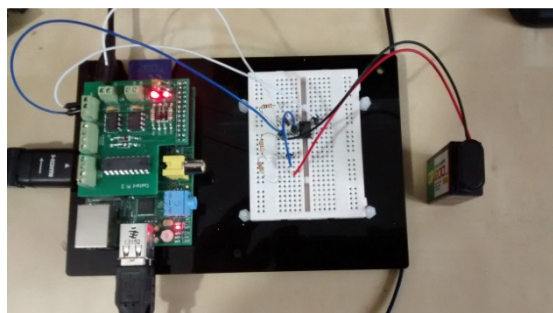
MCP6002 op-amp: The voltage of a PP3 battery is nominally 9V and can be a little bit higher. The maximum input into the A to D convertor that we are planning to use is 3.3V. For this reason we have to convert the actual battery voltage into a lower voltage. We are going to use one of the op-amps from the chip shown below for this.



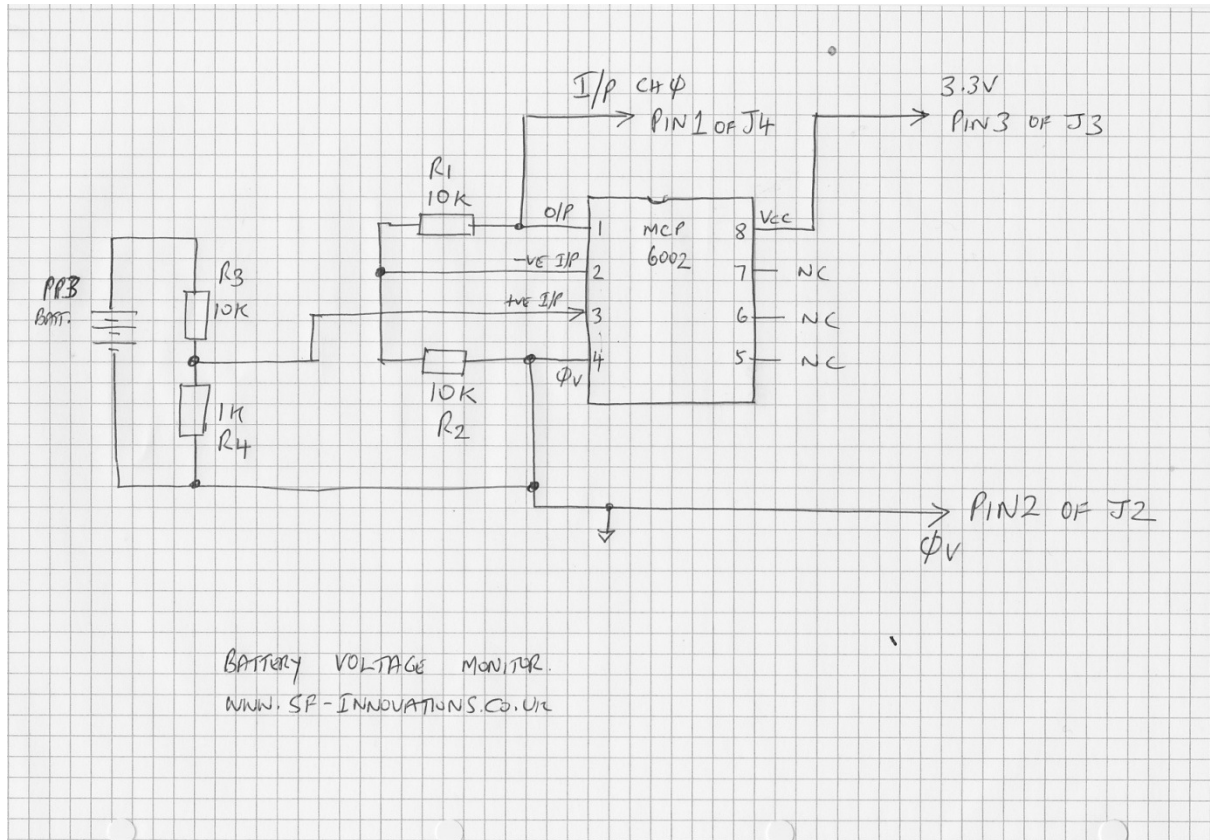
Custard Pi 2: The Raspberry Pi GPIO does not have any analogue inputs so we are going to use the Custard Pi 2 which has 2 analogue inputs.



For ease of assembly, this project uses a Custard Pi base and a prototyping board. The assembled hardware is shown below.



We are going to use the 3.3V and 0V connections on the Custard Pi 2 to power the MCP6002. The circuit diagram is shown below along with a detailed description.



The input from the battery is taken to the +ve input and the -ve input has resistor R1 to the output another resistor R2 to 0V. In this configuration the gain is given by the following equation.

$$\text{Gain} = 1 + R1/R2.$$

Resistors R1 and R2 provide a gain of 2 through the op-amp as they are both 10k. The main purpose of the op-amp is to buffer the signal going to the ADC (Analogue to Digital Converter) input of the Custard Pi 2. ie the input resistance of the ADC has no effect on the voltage of the measuring circuits.

As the battery voltage (at 9V) is much higher than the maximum of 3.3V expected by the ADC, we have to attenuate the signal. This is done by resistors R3 and R4. The attenuation provided is given by the equation:

$$\text{Attenuation} = R4/(R3+R4) = 1/11 = 0.091$$

So the battery voltage is first attenuated by 0.091 and then amplified by 2. To get to the battery voltage from the measured voltage we have to multiply by (1/0.091 or 10.99) and then divide by 2.

Software

To make life easier, we are going to use a routine called `cp2adc.py` to read the voltage from one of the 2 channels available on the Custard Pi 2. This has been provided by SF Innovations and is available on their website under the 'Downloads' tab. A full listing of `cp2adc.py` is also provided in appendix A.

By using the function 'readchannel0()' the program for reading voltage is very simple. The Python code below reads the voltage on channel 0, converts it to the true battery voltage and prints the result.

```
import RPi.GPIO as GPIO
from time import sleep
import cpi2adc
GPIO.setmode(GPIO.BOARD)

read0=cpi2adc.readchannel0
read1=cpi2adc.readchannel1

while True:
    V=read0()
    V=V*10.99
    V=V/2
    V= round (V,2)
    print "Battery Voltage=", V
    print
    sleep (1)
```

(Download this code from here: <http://www.sf-innovations.co.uk/downloads>)

The above program uses the routine 'readchannel0' in cpi2adc to read in the modified battery voltage. This value is then multiplied by 10.99 to compensate for the attenuation and divided by 2 to offset the gain introduced by the op-amp. This is then printed to the screen.

Ideas for advanced projects

When the battery voltage drops below a pre-determined level, send out an alert e-mail.

Modify the circuit and software to check the voltage of two PP3 batteries in series at a total voltage of 18V.

Notes:

The Custard Pi 2 and Custard Base are available from amazon.co.uk and directly from SF Innovations. The code presented here can be downloaded from the SF Innovations website.

Appendix A - cpi2adc

The functions 'readchannel0()' and 'readchannel1()' in this routine return a value in volts for input to channel 0 and channel 1.

(Download this code from here: <http://www.sf-innovations.co.uk/downloads>)

```
#!/usr/bin/env python
#program to read analogue voltage on Custard Pi 2
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)

GPIO.setup(26, GPIO.OUT)    #pin26 is chip select 1
GPIO.setup(23, GPIO.OUT)    #pin23 is clock
GPIO.setup(19, GPIO.OUT)    #pin19 is data out
GPIO.setup(24, GPIO.OUT)    #pin24 is chip select 0
GPIO.setup(21, GPIO.IN)     #pin21 is data in
```

```

#set pins to default state
GPIO.output(24, True)
GPIO.output(26, True)
GPIO.output(23, False)
GPIO.output(19, True)

#set up data for ADC chip
word5= [1, 1, 0, 1, 1]
word6= [1, 1, 1, 1, 1]

def readchannel0():
    #reads analogue voltage from channel 0

    GPIO.output(24, False) #select channel 0
    anip=0 #initialise variable

    #set up channel 0
    for x in range (0,5):
        GPIO.output(19, word5[x])
        time.sleep(0.01)
        GPIO.output(23, True)
        time.sleep(0.01)
        GPIO.output(23, False)

    #read analogue voltage
    for x in range (0,12):
        GPIO.output(23,True)
        time.sleep(0.01)
        bit=GPIO.input(21)
        time.sleep(0.01)
        GPIO.output(23,False)
        value=bit*2**(12-x-1)
        anip=anip+value

    GPIO.output(24, True)

    volt = anip*3.3/4096
    return volt

def readchannell1():
    #reads analogue voltage from channel 1

    GPIO.output(24, False) #select channel 1
    anip=0 #initialise variable

    #set up channel 1
    for x in range (0,5):
        GPIO.output(19, word6[x])
        time.sleep(0.01)
        GPIO.output(23, True)
        time.sleep(0.01)
        GPIO.output(23, False)

    #read analogue voltage
    for x in range (0,12):
        GPIO.output(23,True)
        time.sleep(0.01)
        bit=GPIO.input(21)
        time.sleep(0.01)
        GPIO.output(23,False)
        value=bit*2**(12-x-1)
        anip=anip+value

    GPIO.output(24, True)

    volt = anip*3.3/4096
    return volt

```