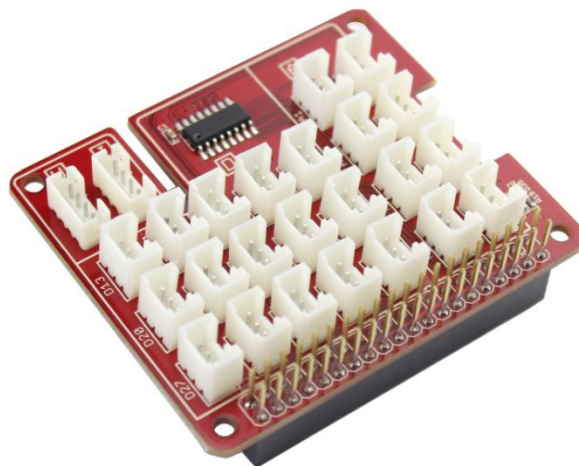
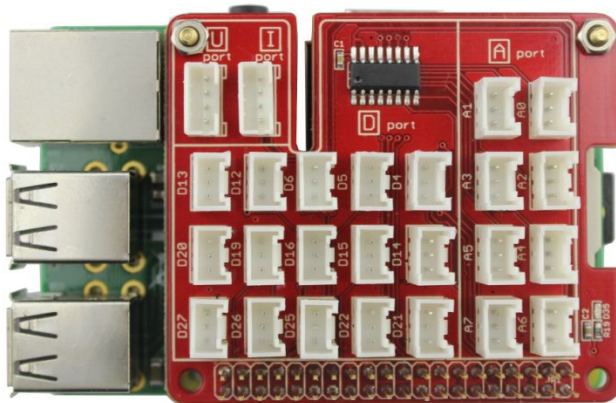


Crowtail- Base Shield for Raspberry Pi

This is a Crowtail base shield for Raspberry Pi with UART/I2C/Analog/Digital interface. All the Crowtail modules can be plugged onto Raspberry Pi through this base shield. Moreover, this shield has an on-board ADC chip so that analog output modules can be used on Raspberry Pi. The ADC chip used is MCP3008. It talks to Raspberry Pi using SPI interface.

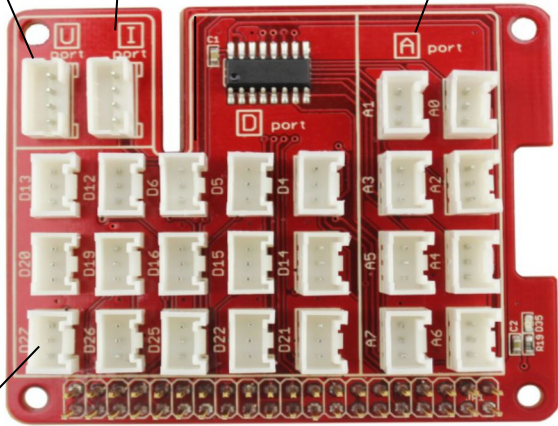


UART
1- RXD
2- TXD
3- V+
4- GND

I2C
1- SCL
2- SDA
3- V+
4- GND

Analog Inputs
A0-A7

Digital GPIO
D4-D6
D12-D27



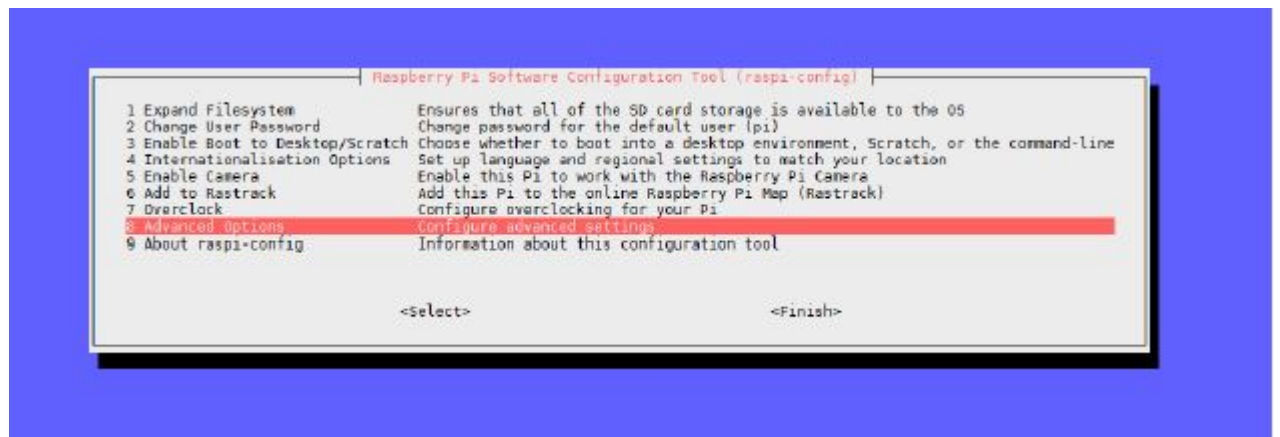
Control the analog inputs through the MCP3008

Installation of the needed modules

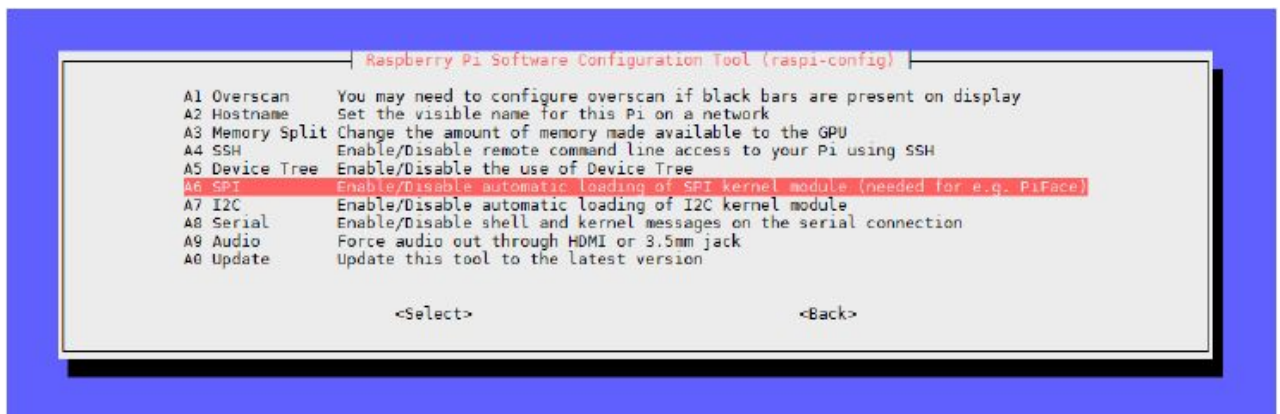
An actual Raspbian OS-Image (Debian Wheezy) is a good choice. First of all we've to activate the SPI-Interface of the Raspberry Pi, to make the communication with the ADC of the shield possible. For this, we're starting with the following command:

```
sudo raspi-config
```

Choose "Advanced Options" when you see this window,



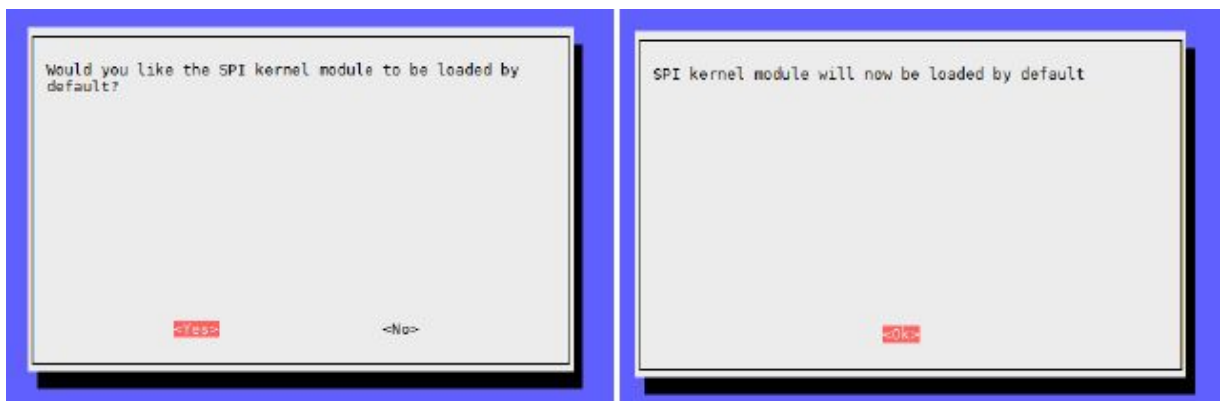
Then we click "A6 SPI"



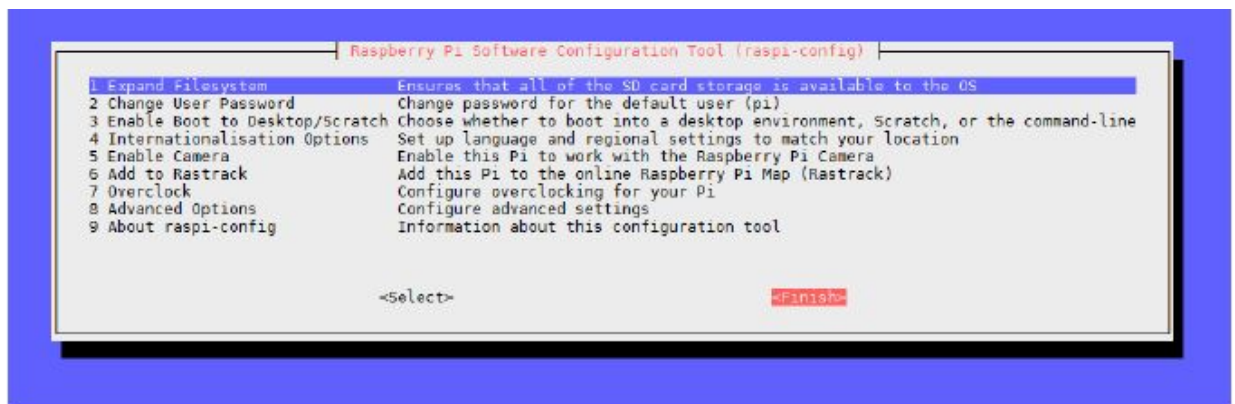
The next two windows have to be accepted with "Yes" and "OK"



Also the next two ones...



Finally we click "Finish" to close this Configuration Tool...



...and reboot the Raspberry Pi with the following command:

```
sudo reboot
```

After the reboot, we need to load and install the needed Drivers and Modules. The Raspberry Pi need to be connected to the Internet first in this step- put the following commands in the console of the Raspberry Pi and confirm each with press on [Enter]. :

```
sudo apt-get update
```

```
sudo apt-get install python-imaging python-imaging-tk python-pip python-dev git
```

```
sudo pip install spidev
```

```
sudo pip install wiringpi
```

At last you have to reboot it again:

```
sudo reboot
```

Python-Example for using the MCP3008 ADC Controller - testadc.py

```

import spidev
import time
import sys

spi = spidev.SpiDev()
spi.open(0,0)

def readadc(adcnum):
    if adcnum >7 or adcnum <0:
        return -1
    r = spi.xfer2([1,8+adcnum <<4,0])
    adcout = ((r[1] &3) <<8)+r[2]
    return adcout

while True:
    if len(sys.argv) >1:
        for i in range(len(sys.argv)):
            if i == 0:
                print "_____ \n"
            else:
                adc_channel = int(sys.argv[i])
                print "Channel " + str(adc_channel)
                value=readadc(adc_channel)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print " "
                print "_____ \n"
                time.sleep(1.5)
            else:
                print "_____ \n"
                print "Channel 0"
                value=readadc(0)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 1"
                value=readadc(1)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 2"
                value=readadc(2)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 3"
                value=readadc(3)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 4"
                value=readadc(4)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 5"
                value=readadc(5)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 6"
                value=readadc(6)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "Channel 7"
                value=readadc(7)
                volts=(value*3.3)/1024
                print("%4d/1023 => %5.3f V" % (value, volts))
                print "_____ \n"
                time.sleep(1.5)

```

The file "testadc.py" shows how to measure analog voltage level with a python script: with the function readadc() and a specified Channel (0-7) you can read out the actual value. Create a new file with the name "testadc.py" and copy the script shown above in this file (please make sure that you don't miss any spaces).

The script can be used in two different ways now:

Output the values of all ADC-Channels simultaneously:

```
sudo python testadc.py
```

This command causes that the values of all ADC-Channels be read out and displayed every 1.5s.

```
Channel 0
 513/1023 => 1.653 V
Channel 1
 519/1023 => 1.673 V
Channel 2
  0/1023 => 0.000 V
Channel 3
  0/1023 => 0.000 V
Channel 4
  0/1023 => 0.000 V
Channel 5
  0/1023 => 0.000 V
Channel 6
  0/1023 => 0.000 V
Channel 7
  0/1023 => 0.000 V
```

Output of values from specific Channels:

```
sudo python testadc.py 3 7
```

You can also read out the values from individual channels – specify them with their channel number directly after the command, and you should to click the space key to separate this two commands.

In this example this would be the channels [3] and [7]

```
Channel 3
 518/1023 => 1.669 V

Channel 7
  1/1023 => 0.003 V
```