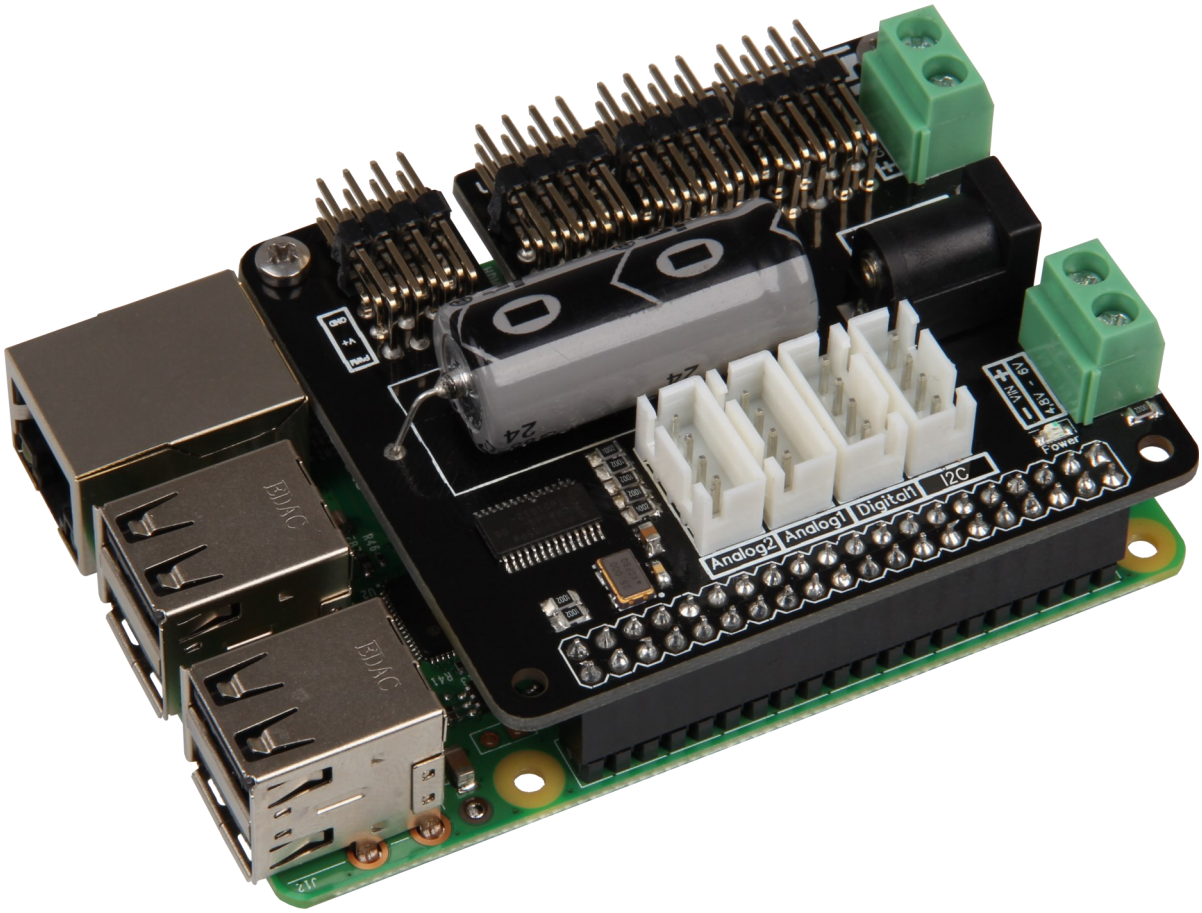
MotoPi



Index

1. Introduction

2. Technical specification & security information

3. Setting up the Raspberry Pi

4. Installation

5. Control of the additional connections

6. Code-Exampe for using the digital connections

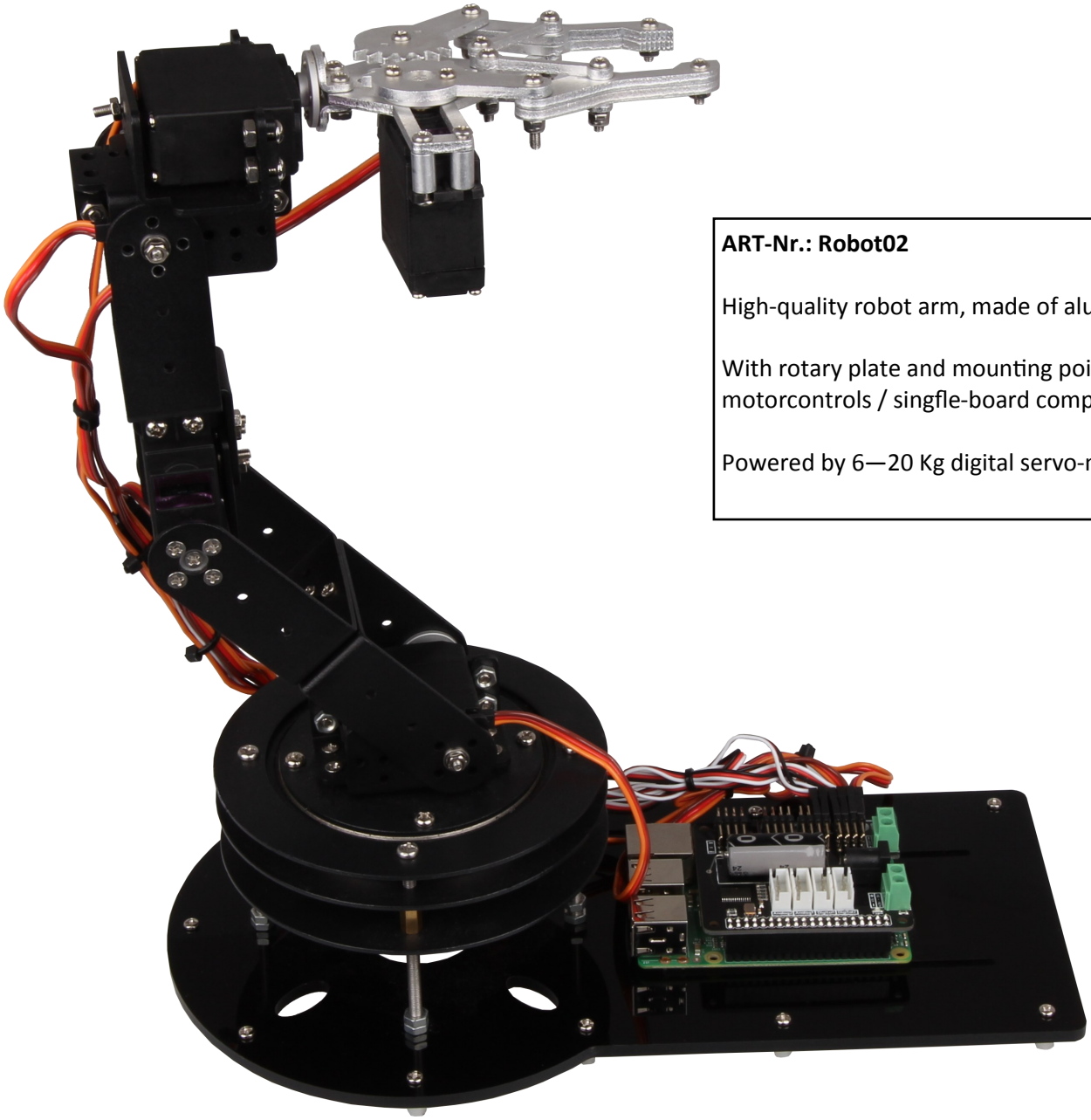
7. Code-Example for using the analog connections

Dear customer,
thank you for purchasing our product. Please find our instructions below.

1. Introduction

The MotoPi board is an extension-board for connecting and using up to 16 PWM-controlled-5V-Servomotors. The board can be powered by a voltage between 4,8V and 6V so the optimal power supply is always assured. Also larger projects are no problem with this extension board.

This board is, for example, perfect for controlling the JOY-iT robot arm.



ART-Nr.: Robot02

High-quality robot arm, made of aluminium

With rotary plate and mounting point for motorcontrols / singfle-board computer.

Powered by 6—20 Kg digital servo-motors.

Image 1: Robot02

2. Technical specification & security information

The MotoPi-Extensionboard is equipped with 16 channels for 5V-servomotors and with an optinal connection for an additional capacitor.

The board has also 4 analog, 2 digital and one I2C connection possibility.
The power supply can be established by a 5V coaxial power connector or by a power supply between 4,8V and 6V.

The MotoPi-Board is also equipped with an additional crystal oscillator which makes the frequencys as precise as possible and the discrepancy as low as possible.

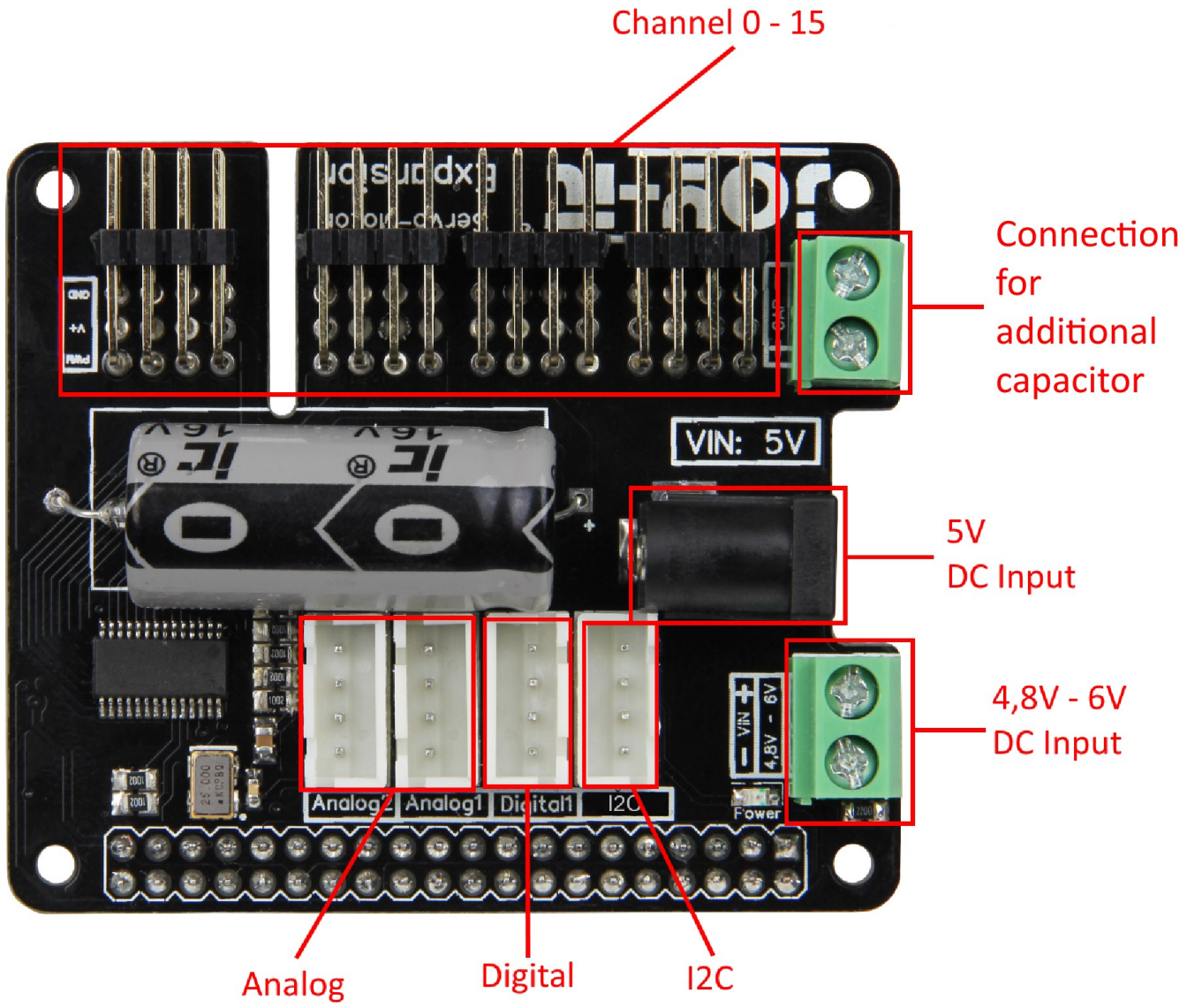


Image 2: Board-Description

The channel-numbers are written on the backside of the board.

The 3 PINs of each channel are, from bottom to top, ground line, voltage line and signal line [GND | V+ | PWM].

Simply put the board onto the GPIO-PINs of your Raspberry Pi and the cables of your 5V-Servo-Motors on the Channel-PINs.

An additional power supply, by cable or a 5V coaxial power connector, is **required**.

To prevent sudden voltage drops, a capacitor is mounted on the board.
If this capacitor is insufficient in special cases, you are able to connect another capacitor in parallel with the intended connection.

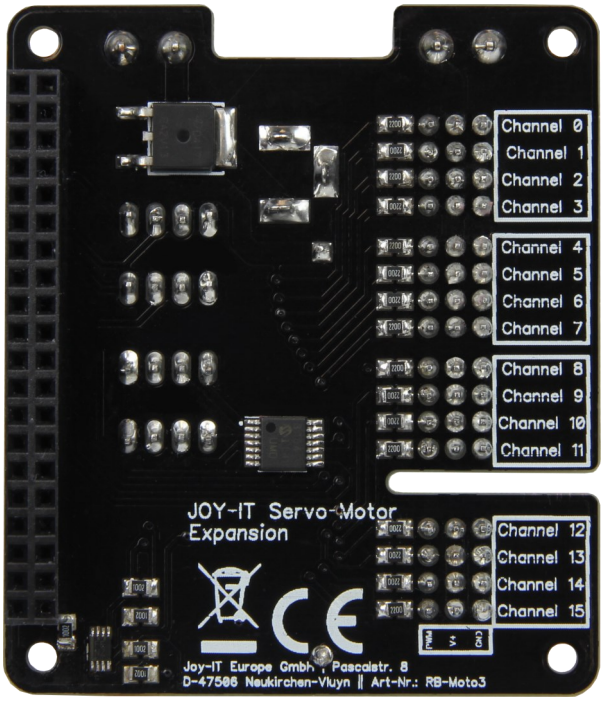


Image 3: Board bac kside

The used power supply has to be adequate for the performance of the motors.
A weak power supply is often recognizable by stuttering motors.
The motors should always perform a smooth motion.

We recommend to use our 4.8A power supply *RB-Netzteil2* which provides 24W continous output.
After connecting the board to your Raspberry Pi, an additional power supply and the servo-motors, the board is ready to use.

Security information:

To protect against polarity reversal, please note the markings on the inputs of the board (+ and - symbol).
Polarity reversal can damage the board, the connected Raspberry Pi and other periphery.
The connected motors, and the generated movement, can pose a risk.
We recommend to remain at a safe distance and take actions so that noone can encounter with moving parts.
This applies in particular for children.

3. Installation des Raspberry Pis

If you are already using the current version if the Raspbian system, you can skip this step and continue with the next one.

You can download the current Raspbian Image from the Rasperry Pi Website.

With the „Win32 Disk Imager“-utility you can copy the downloaded image to your SD-Card.

Select, as seen in the image below, the image and the device. Then you can start the writing process with *Write*.

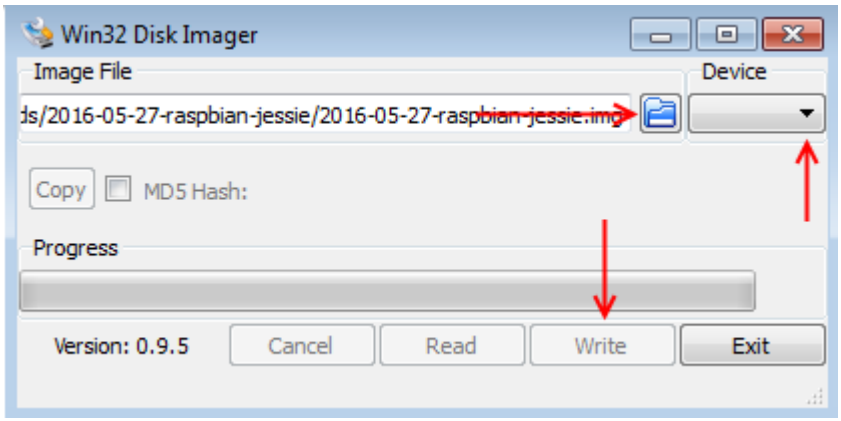


Image 4: Win32 Disk Imager

If this step is completed, you can put the SD-Card into you Raspberry Pi and continue.

4. Installation

As seen in *Image* 2, the board also offers 4 analog, 2 digital and one I2C channels.

To be able to use these channels, you need to activate the SPI-Configration at first.
Therefore you need to open up the Raspberry Pis configuration menu.

```
sudo raspi-config
```

In the opened window, navigate to the menu *Interfacing Options*.

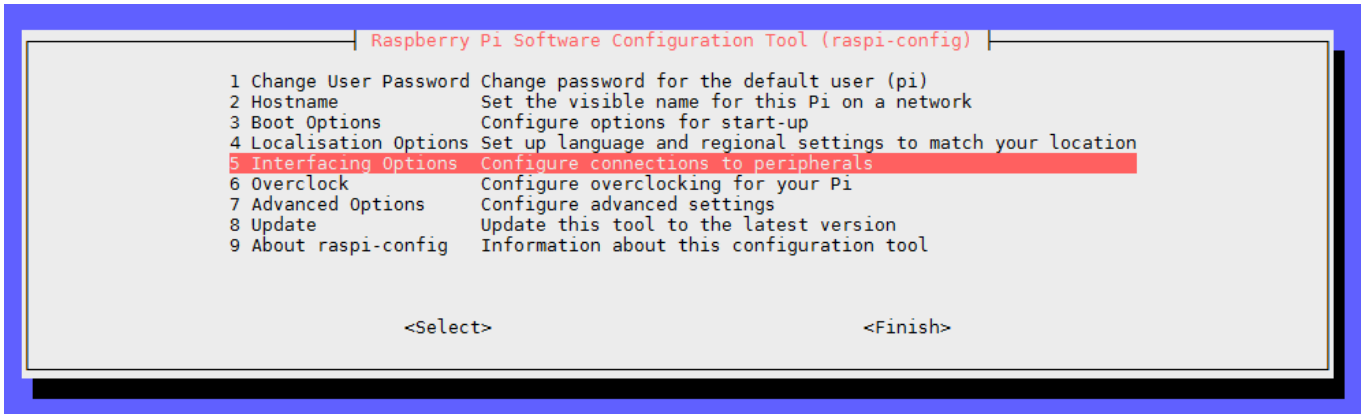


Image 5: Raspi-Config

Activate the *SPI* option.

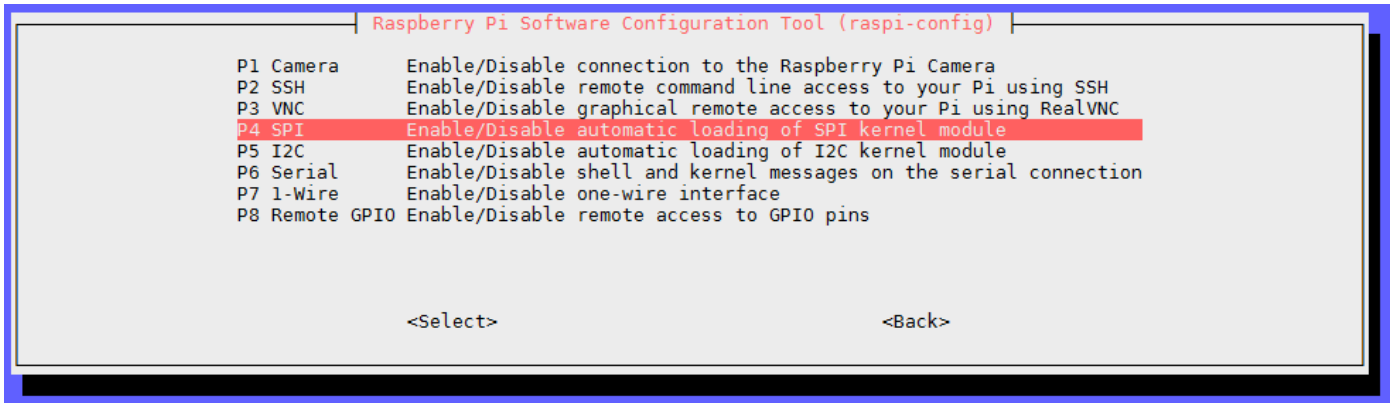


Image 6: Advanced Options

Confirm the next windows with *Yes* or *Ok*.

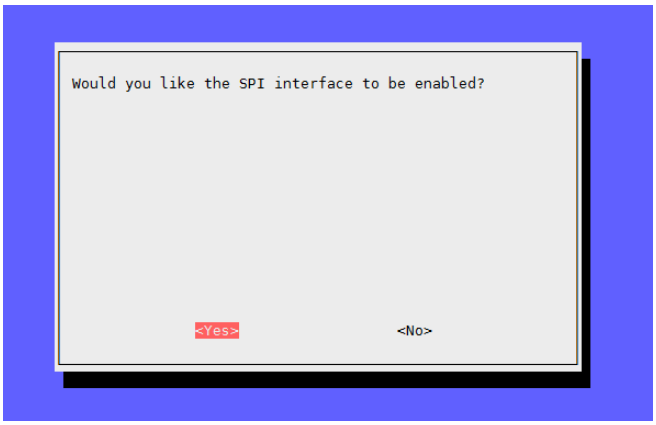


Image 7 & 8: Confirmation SPI

Image 9 & 10: Confirmation load-by-default

Leave the configuration-menu with *Finish* and restart your Raspberry Pi.

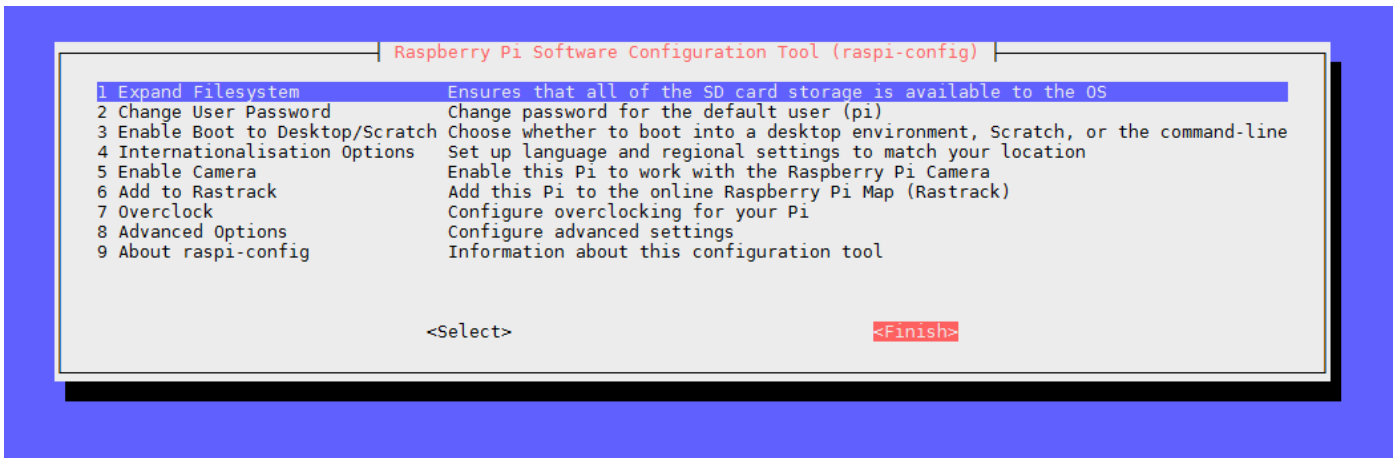


Image 11: leave Raspi-Config

```
sudo reboot
```

Because our MotoPi-board is equipped with an additional crystal oscillator, a special library is needed.Da
This library is based on the Adafruit_PCA9685 Python-library but is adjusted specially for our board.

We recommend to exclusively use our own, adjusted library.
You can download our library here.

Copy the extracted library onto your Raspberry Pi and navigate in the terminal to this folder.

You can install the library with the following command:

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

Next, navigate, as instructed below, to the *examples* folder and start the example-code.

```
cd examples
```

This example demonstrates the basic usage of the motor-control and will move a motor, which is

```
sudo python simpletest.py
```

connected to the first channel.

5. Control of the additional connections

Execute the following terminal-commands:

```
sudo apt-get update
```

```
sudo pip install spidev
```

```
Sudo pip install wiringpi
```

Another restart is now required.

```
sudo reboot
```

All connections are ready to use.
Please note that, at the digital connection, the first PIN is refered to GPIO Port 27 and the second PIN is refered to GPIO Port 22.

6. Code-Example for using the digital connections

You can find a short example for controlling the digital conenctions below.
We used a *LK-Button1* and a *LK-Cable-20* for demonstration from our *LinkerKit-Series*.

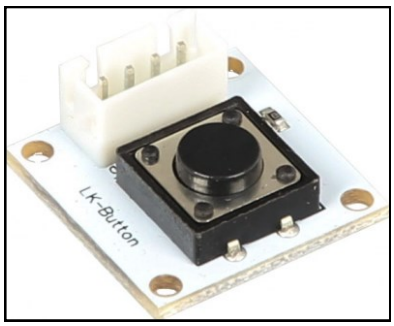


Image 12 & 13: LK-Button1 & LK-Cable-20

```
import RPi.GPIO as GPIO
from time import sleep
#Initialisiere Button auf Digital-PIN 22
button = 22
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(button, GPIO.IN, pull_up_down=GPIO.PUD_UP)
while True:
      if GPIO.input(button) == GPIO.HIGH:
            #Do something
            print "I do something"
      else:
            #Do something else
            print "I do something else"
```

7. Code-Example for using the analog connections

```
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 "_____________________________________________\n"
            time.sleep(1.5)
```