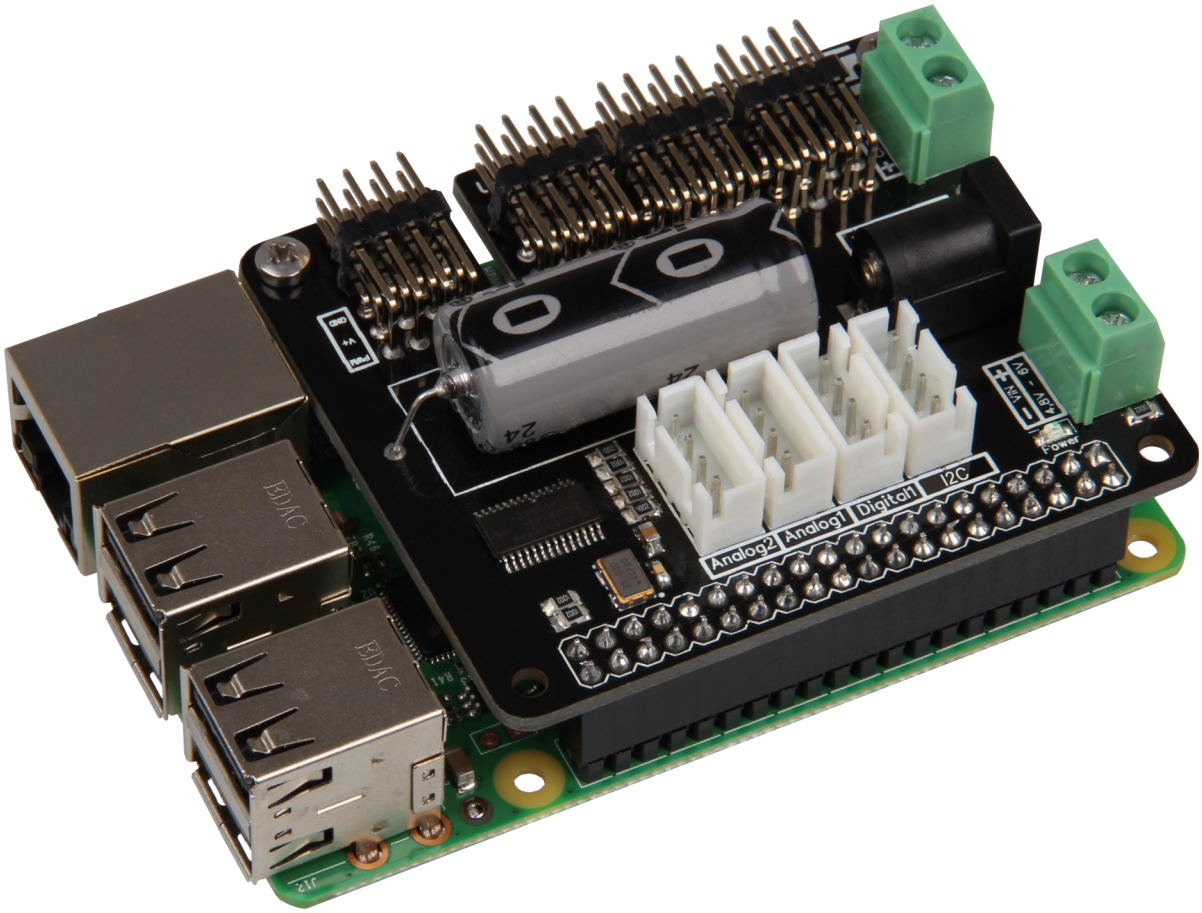
JOY-IT

MOTOPI

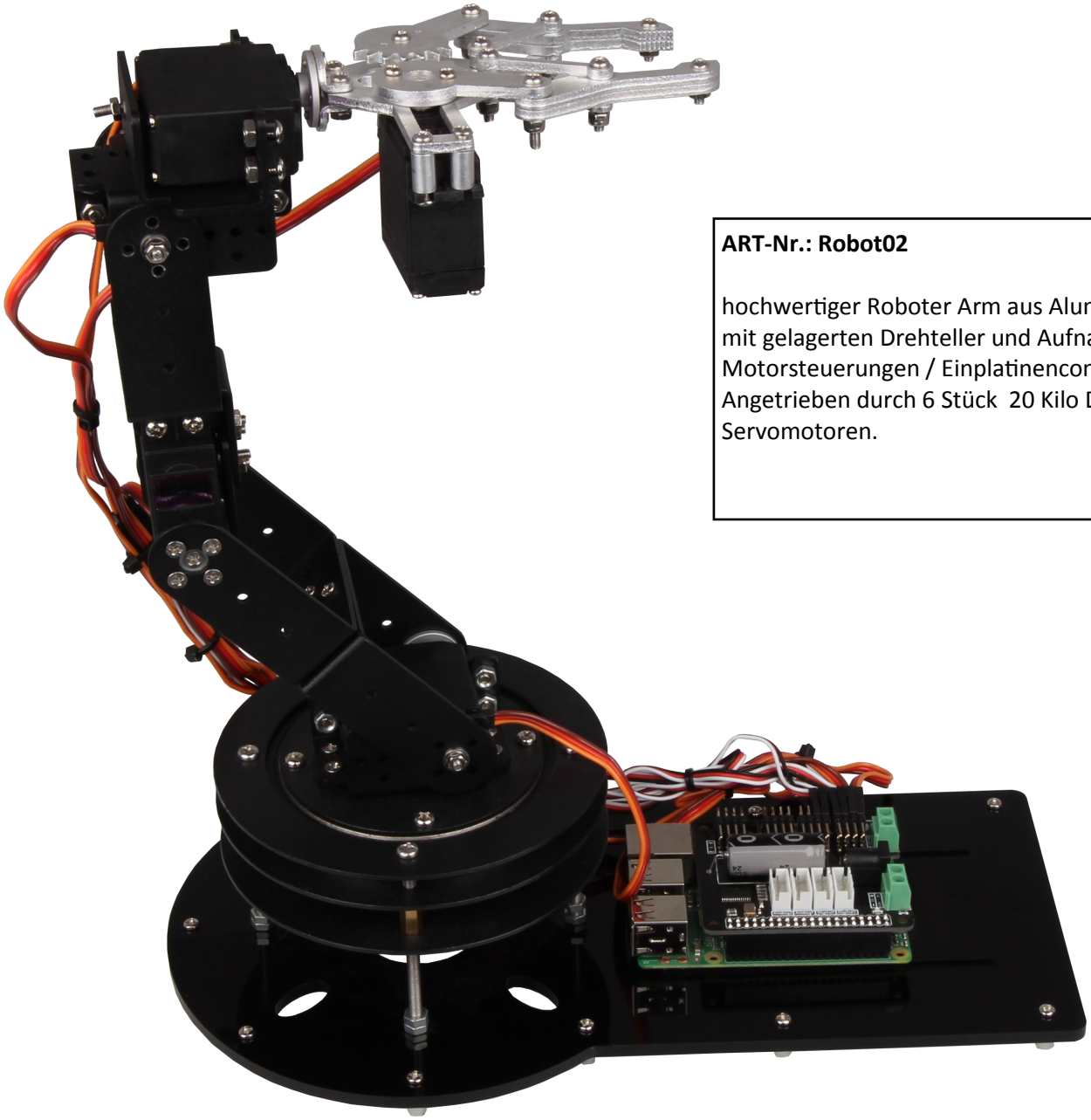


Index

Sehr geehrter Kunde,
vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden haben wir aufgelistet, was bei der Inbetriebnahme zu beachten ist:

1. Einführung

Die MotoPi Platine ist eine Erweiterungsplatine zur Ansteuerung und Verwendung von bis zu 16 PWM-gesteuerten 5V-Servomotoren.
Die Platine kann zusätzlich mit einer Spannung von 4,8V - 6V versorgt werden, sodass eine optimale Versorgung der Motoren stets gewährleistet ist und somit auch größere Projekte mit ausreichend Strom beliefert werden können.
Er ist z.B. perfekt geeignet für die Steuerung des JOY-iT Roboter Arms



ART-Nr.: Robot02

hochwertiger Roboter Arm aus Aluminium mit gelagerten Drehteller und Aufnahme für Motorsteuerungen / Einplatinencomputer. Angetrieben durch 6 Stück 20 Kilo Digital Servomotoren.

Abb. 1: Robot02

2. Technische Daten und Sicherheitshinweise

Die MotoPi-Erweiterungsplatine ist ausgestattet mit 16 Kanälen für 5V-Servomotoren, sowie mit einer Anschlussmöglichkeit für einen zusätzlichen Kondensator.
Außerdem verfügt die Platine über 4 analoge, 2 digitale und eine I2C Anschlussmöglichkeit.
Die Stromversorgung erfolgt, wahlweise, über einen 5V Hohlstecker oder über eine Spannungsversorgung zwischen 4,8V und 6V.

Die MotoPi-Platine ist außerdem mit einem zusätzlichen Quarzoszillator ausgestattet, um die Frequenzen so präzise wie möglich und die Abweichungen möglichst gering zu halten.

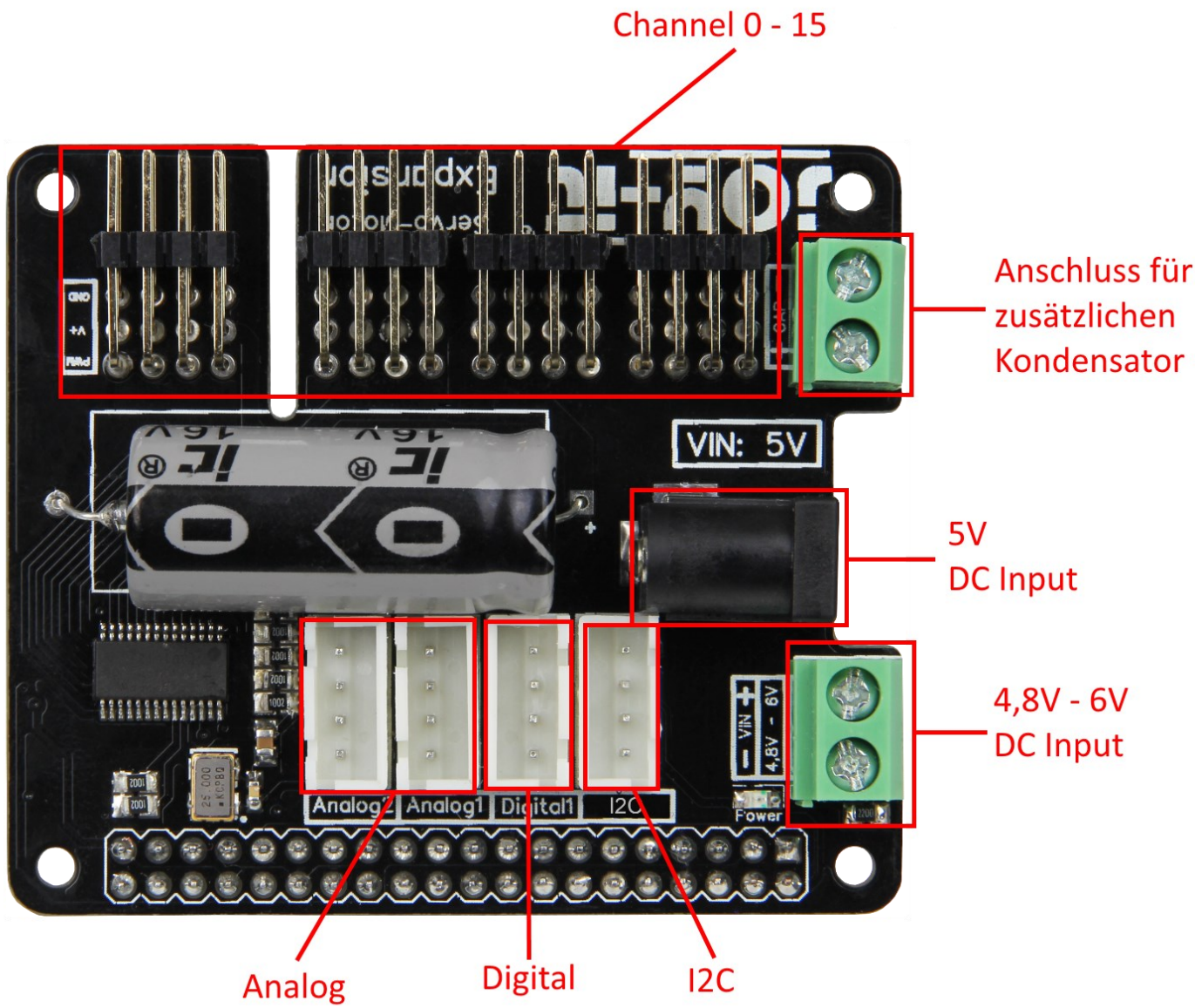


Abb. 2: Platinenbeschreibung

Auf der Unterseite der Platine können die Channelnummerierungen abgelesen werden.

Die 3 PINs des jeweiligen Channels sind, von unten nach oben gesehen, für den Anschluss der Masseleitung, der Spannungsleitung und der Signalleitung [GND| V+ | PWM].

Stecken Sie die Platine einfach auf die GPIO-PINs Ihres Raspberry Pi's auf und die Kabel Ihrer 5V-Servomotoren an die Channel-PINs.

Eine zusätzliche Spannungsversorgung über ein Kabel oder einen 5V Hohlstecker ist für den Betrieb **zwingend** erforderlich.

Um plötzlichen Spannungsabfällen vorzubeugen ist auf der Platine bereits ein Kondensator angebracht.
Sollte dies jedoch in speziellen Ausnahmefällen nicht genügen, so kann ein weiterer Kondensator über den dafür vorgesehenen Anschluss parallel geschaltet werden.

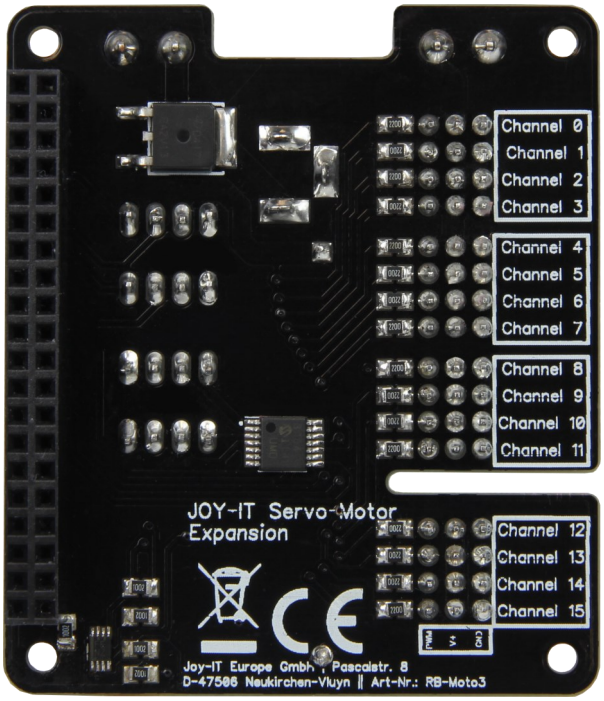


Abb. 3: Platinenrückseite

Das von Ihnen eingesetzte Netzteil muss auf die angeschlossenen Motoren von der Leistung her abgestimmt sein, zu schwach dimensionierte Netzteile erkennt man häufig da dran, das die Motoren ruckeln statt eine fließende Bewegung durchzuführen.
Wir empfehlen die Verwendung unseres 4.8A Netzteil *RB-Netzteil2* das 24W Dauerleistung zur Verfügung stellt.

Nach dem Aufstecken der Erweiterungsplatine auf den Raspberry Pi, dem Anschließen einer zusätzlichen Stromversorgung und dem Anschließen von Servomotoren, ist die Platine einsatzbereit.

Sicherheitshinweis:

Um eine Verpolung zu vermeiden, beachten Sie bitte die auf der Platine angebrachten Kennzeichnungen der jeweiligen Eingänge (+ und - Symbol).
Eine Verpolung kann zu Schäden an der Platine, dem angeschlossenen Raspberry Pi und weiterer Peripherie führen!
Die von Ihnen angeschlossenen Motoren, und die durch sie erzeugte Bewegung,
können eine Gefahr darstellen.
Wir empfehlen daher ausreichenden Sicherheitsabstand zu halten bzw. Maßnahmen zu treffen,
damit niemand mit bewegten Teilen in Berührung kommen kann.
Dies gilt insbesondere für Kinder.

3. Installation des Raspberry Pis

Sollten Sie auf Ihrem Raspberry Pi bereits ein aktuelle Raspbian Version verwenden, so können Sie diesen Schritt überspringen und sofort mit dem nächsten Schritt fortfahren.

Das aktuellste Image des Betriebssystems können Sie auf der Rasperry Pi Website herunterladen.

Mit Hilfe des „Win32 Disk Imager“-Programms können Sie das heruntergeladene Image auf Ihre SD-Karte kopieren.
Wählen Sie dafür, wie in der unten stehenden Abbildung aufgezeigt, das Image und das zu beschreibende Gerät aus.
Anschließend kann der Schreibvorgang mit *Write* gestartet werden.

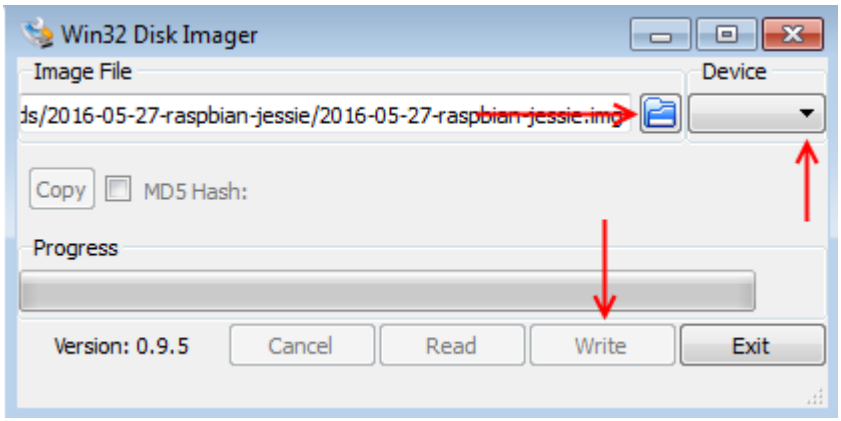


Abb. 4: Win32 Disk Imager

Ist dieser Schritt abgeschlossen, so können Sie die beschriebene SD-Karte in Ihren Raspberry Pi einlegen und fortfahren.

4. Installation des Moduls

Um die Platine ordnungsgemäß verwenden zu können, sollten Sie zunächst, wie untenstehend beschrieben, die I2C Funktion an Ihrem Raspberry Pi aktivieren.

Öffnen Sie daher zunächst das Raspberry Pi Konfigurationsmenü.

```
sudo raspi-config
```

In dem sich nun öffnenden Fenster navigieren Sie in das Menü *Interfacing Options*.

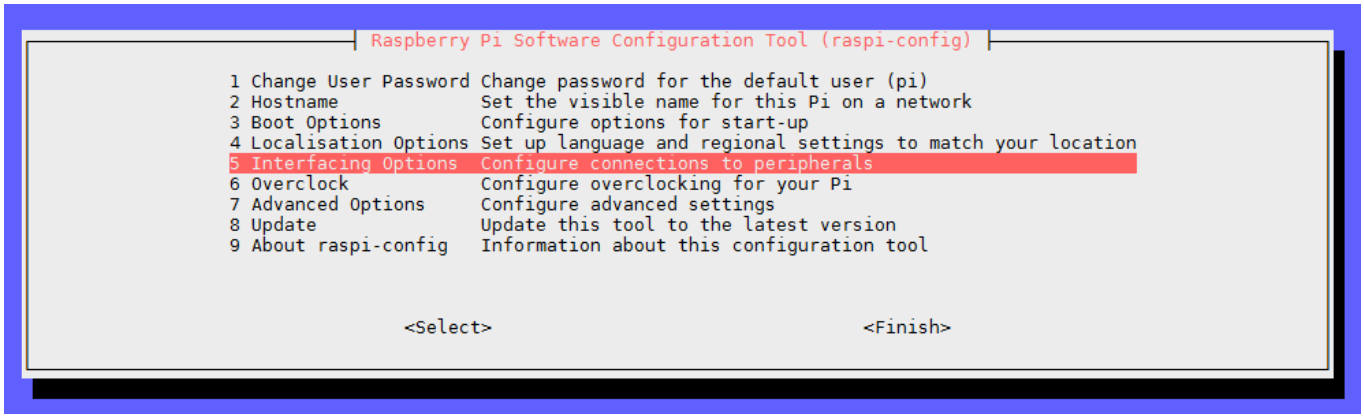


Abb. 5: Raspi-Config

Hier aktivieren Sie die Option *SPI*.

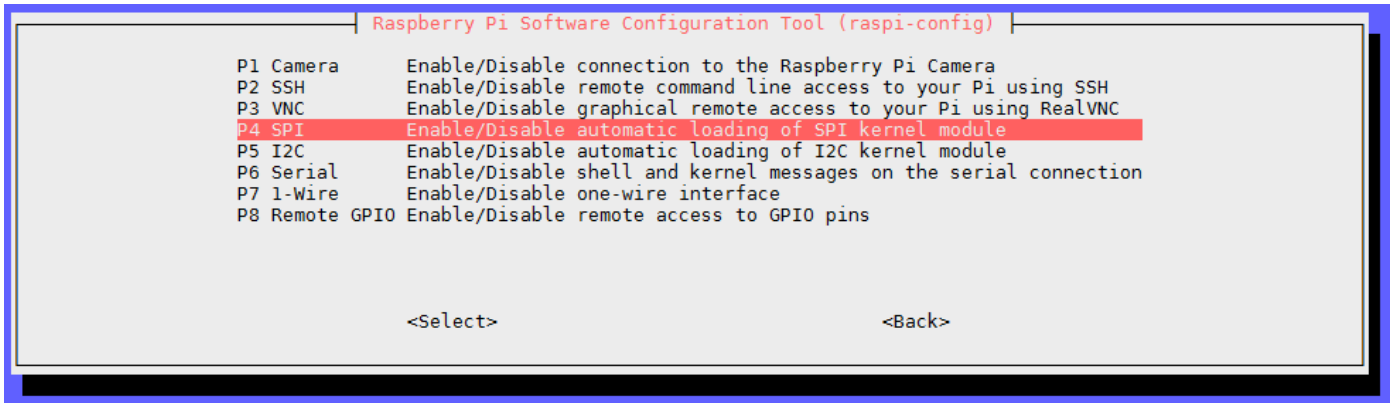


Abb. 6: Advanced Options

Die nächsten sich öffnenden Fenster bestätigen Sie bitte mit *Yes* bzw. *Ok*.

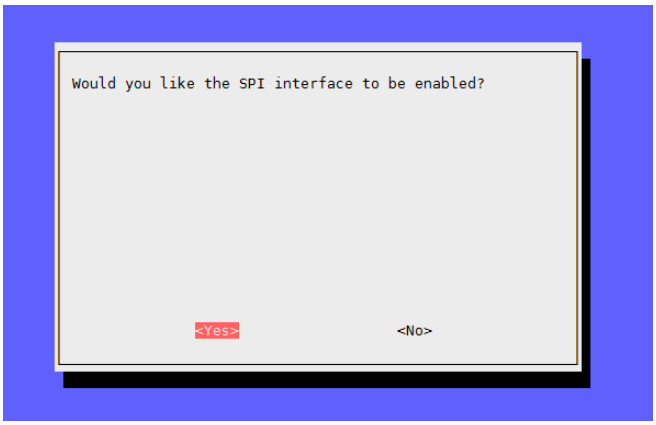


Abb. 7 & 8: Bestätigung SPI

Abb. 9 & 10: Bestätigung load-by-default

Verlassen Sie das Konfigurationsmenü nun mit *Finish* und starten Sie ihren Raspberry Pi neu.

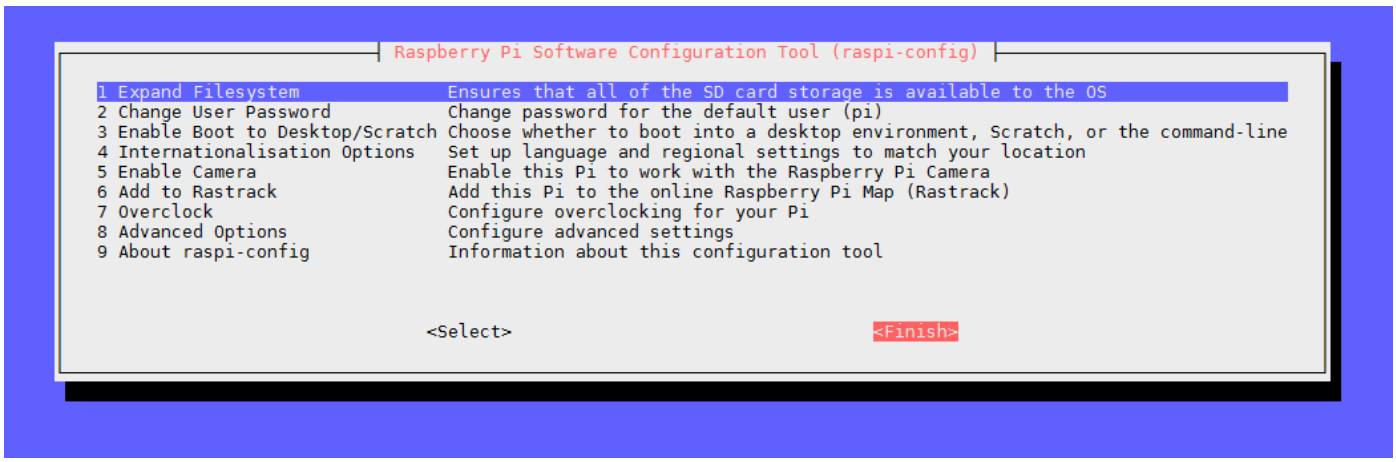


Abb. 11: Raspi-Config verlassen

```
sudo reboot
```

Da unsere MotoPi-Platine mit einem zusätzlichen Quarzoszillator ausgestattet ist, ist eine besondere Programmbibliothek erforderlich.
Diese basiert auf der Adafruit_PCA9685 Python-Bibliothek, ist allerdings speziell auf unsere Platine angepasst.

Wir empfehlen unbedingt ausschließlich unsere eigene, abgestimmte, Bibliothek zu verwenden.
Die angepasste Bibliothek können Sie hier herunterladen.

Kopieren Sie die entpackte Bibliothek bitte vollständig auf Ihren Raspberry Pi und navigieren Sie im Terminal in diesen Ordner.

Mit dem folgenden Befehl können Sie die Bibliothek dann installieren:

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

Navigieren Sie nun, wie folgend beschrieben, in den *examples* Ordner und starten Sie den Beispielcode.

```
cd examples
```

```
sudo python simpletest.py
```

Dieser Funktionstest zeigt die grundlegende Funktionsweise der Motorenansteuerung und wird einen Motor, der am ersten Kanal angeschlossen ist, wiederholt hin und her bewegen.

5. Ansteuerung der zusätzlichen Anschlüsse

Nach dem Neustart müssen die benötigten Treiber und Module installiert werden.
Führen Sie daher im Terminal folgende Befehle aus.

```
sudo apt-get update
```

```
sudo pip install spidev
```

```
Sudo pip install wiringpi
```

Ein erneutet Neustart ist nun erforderlich.

```
sudo reboot
```

Nach dem Neustart sind die Anschlüsse einsatzbereit.
Bitte beachten Sie, dass beim digitalen Anschluss der erste PIN auf den GPIO Port 27 und der zweite PIN auf den GPIO Port 22 verweist.

6. Code-Beispiel zur Verwendung der digitalen Anschlüsse

Nachfolgend können Sie ein kurzes Anwendungsbeispiel zur Ansteuerung der zusätzlichen Anschlüsse entnehmen.
Zur Demonstration verwenden wir hier einen *LK-Button1* mit einem *LK-Cable-20* aus unserer LinkerKit-Serie.

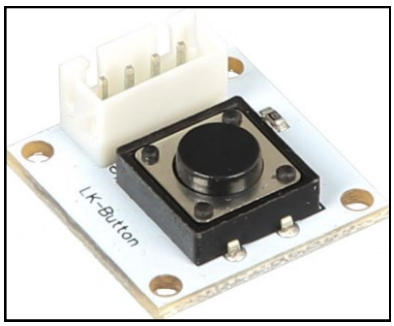


Abb. 12 & 13: LK-Button1 & LK-Cable-20

```
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:
            #Mache etwas
            print "Ich tue etwas"
      else:
            #Mache etwas anderes
            print "Ich tue etwas anderes"
```

7. Code-Beispiel zur Verwendung der analogen Anschlüsse

```
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)
```