

# Raspberry Pi Shield

*ermöglicht die einfachere Entwicklung von Computer basierten Steuerungen auf Basis des Raspberry Pi*

## Inhaltsverzeichnis

Einleitung.....	2
Überblick der Funktionen.....	2
Technische Spezifikationen.....	3
Mechanische Spezifikationen.....	3
Elektrischer Arbeitsbereich.....	3
Prototyp.....	4
Bestückung.....	5
Top-Layer.....	5
Bottom-Layer.....	5
Inbetriebnahme der Hardware.....	6
Flashen des Steuer-PIC.....	6
Testen des Step-Down-Wandler.....	6
Fehlerbehebung.....	6
Testen der Hardware.....	6
Überblick über die Status-LEDs.....	7
PWR.....	7
+5V.....	7
INFO.....	7
STAT.....	7
Inbetriebnahme der Software.....	7
Aktivieren von I2C und SPI.....	7
Aktivieren der I2C-Echtzeituhr.....	7
Aktivieren des LCD-Touch-Displays (MI0283QT-9A).....	9
Neue Anleitung.....	9
Alte Anleitung.....	9
PIC-Firmware.....	11
I2C.....	11
Register.....	11

## Einleitung

Der Raspberry Pi ist ein einfacher Einplatinencomputer, welcher ideal für komplexe Steuerungs-/Regelungsaufgaben im Bereich der Robotik, Hausautomatisierung, und vielen anderen Bereichen ist.

Er zeichnet sich durch einen niedrigen Stromverbrauch in Verbindung mit einer großen Anzahl von Schnittstellen, angefangen von USB und LAN über Bussysteme wie SPI und I2C bis zu einzelnen Digitalen Ausgängen, welche mithilfe von Linux angesprochen werden können.

Die Nutzung der Low-Level Schnittstellen die an einer Stiftleiste herausgeführt sind gestaltet sich aber nicht immer trivial, auch ist die Spannungsversorgung in Robotern nicht für eine zusätzliche Last mit bis zu 1 A ausgelegt. Aus diesem und anderen Gründen soll daher ein Erweiterungsboard für den Raspberry Pi entstehen welches grundlegende Aufgaben übernimmt, die für Systeme in Steuerungssystemen notwendig erscheinen.

## Überblick der Funktionen

Die Grundlegenden Ausstattung, die die Erweiterungsplatine besitzen soll, bzw. bereits besitzt.

- Effizienter Spannungsregler der Spannungen zwischen 7V-25V auf saubere 5V regelt
- Pegelwandler für I2C, um sich mit normaler 5V-Hardware zu verbinden
- Schnittstelle für LCD-Touch-Display auf SPI-Basis, welche später auch für SPI->CAN Konverter benutzt werden kann.
- Echtzeituhr (RTC), weil der Raspberry Pi keine besitzt, diese aber für diverse Anwendungen benötigt werden
- Spannungsüberwachung & Überstromschutz, um auch mit LiPo-Versorgung sicher zu arbeiten (Tiefentladeschutz)
- 8 Digitale Ein/Ausgänge und 8 Analoge Eingänge um ohne zusätzlicher Erweiterungsboard bereits einfache Steuerungen zu realisieren.
- RS232-TTL Schnittstelle
- RN-Standard Steckverbinder mit Verriegelung, für einfache Erweiterbarkeit und Betriebssicherheit.

## Technische Spezifikationen

Müssen überprüft werden!

### Mechanische Spezifikationen

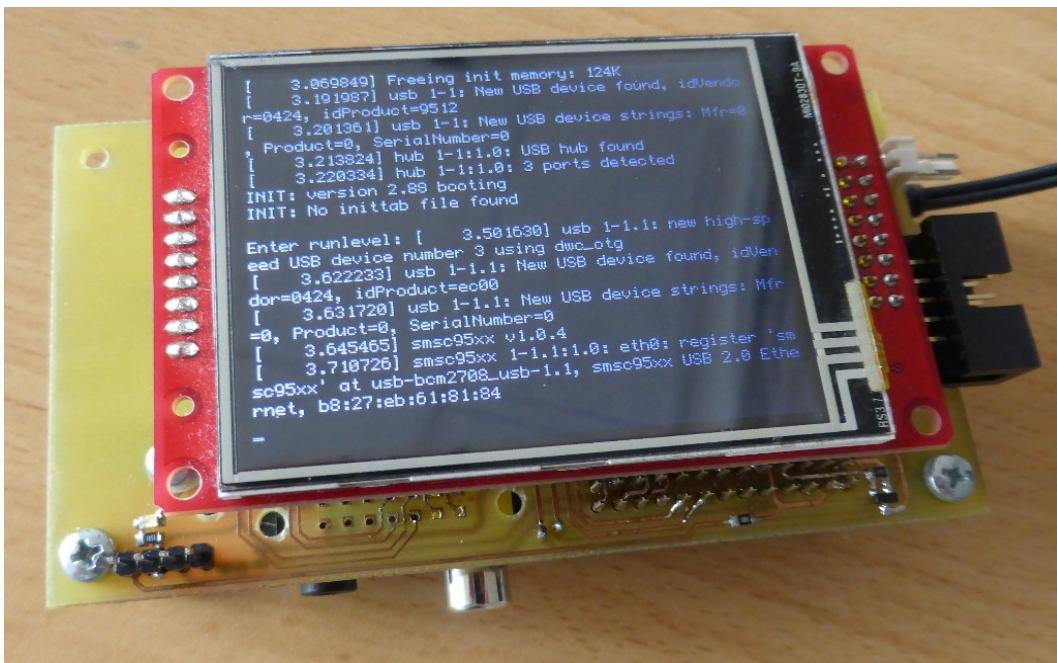
Characteristic		Min	Typ.	Max	Units
Breite		-	60	-	mm
Länge		-	110	-	mm

### Elektrischer Arbeitsbereich

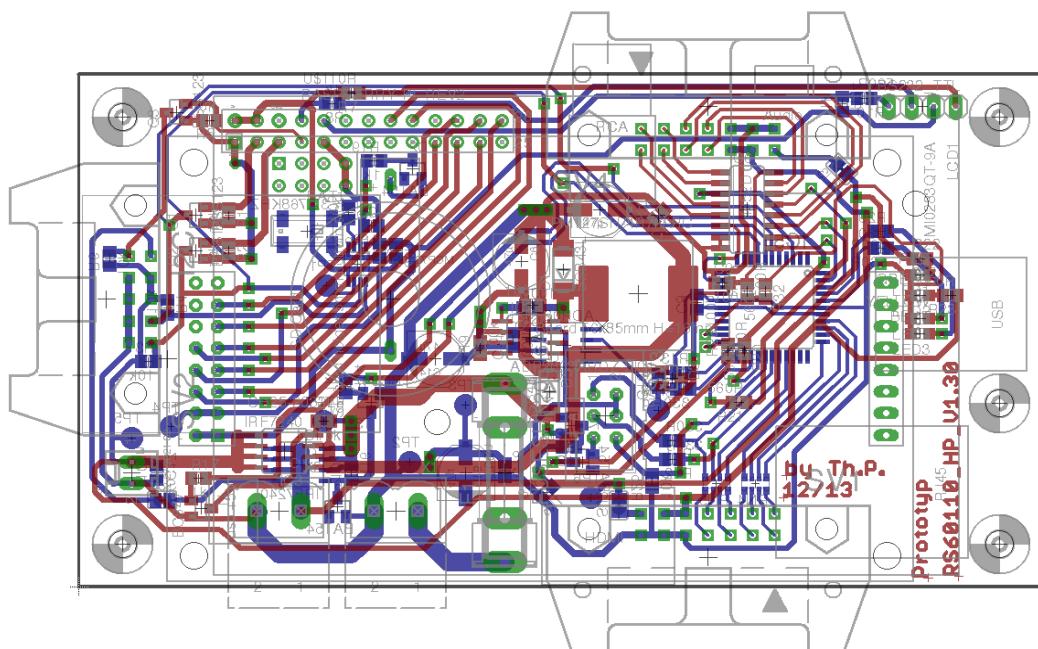
Characteristic	Symbol	Test Conditions	Min	Typ.	Max	Units
Versorgungsspannung	$U_{IN}$		8	12	20	V
Eingangsstrom	$I_{IN}$		-	-	8	A
Leerlaufstrom	$I_{IN0}$	$U_{IN}=12V$	-	20	-	mA
Ausgangsspannung, 5V	$U_{5V}$		4.95	5	5.05	V
Ausgangsstrom, 5V	$I_{5V}$		-	-	2.5	A
Umgebungstemperatur			-10	20	50	°C

## Prototyp

Ich habe bereits einen Prototypen gebaut, der die grundlegende Funktionalität abdeckt, und als Testsystem dient. Das Linux-System bootet, und auch der Steuerungs-Chip auf dem Board arbeitet einwandfrei. Bis zu diesem Stand musste ich aber viele Probleme lösen, und deshalb wird bald eine neue Platine entstehen in der diese Probleme nicht mehr existent sind.

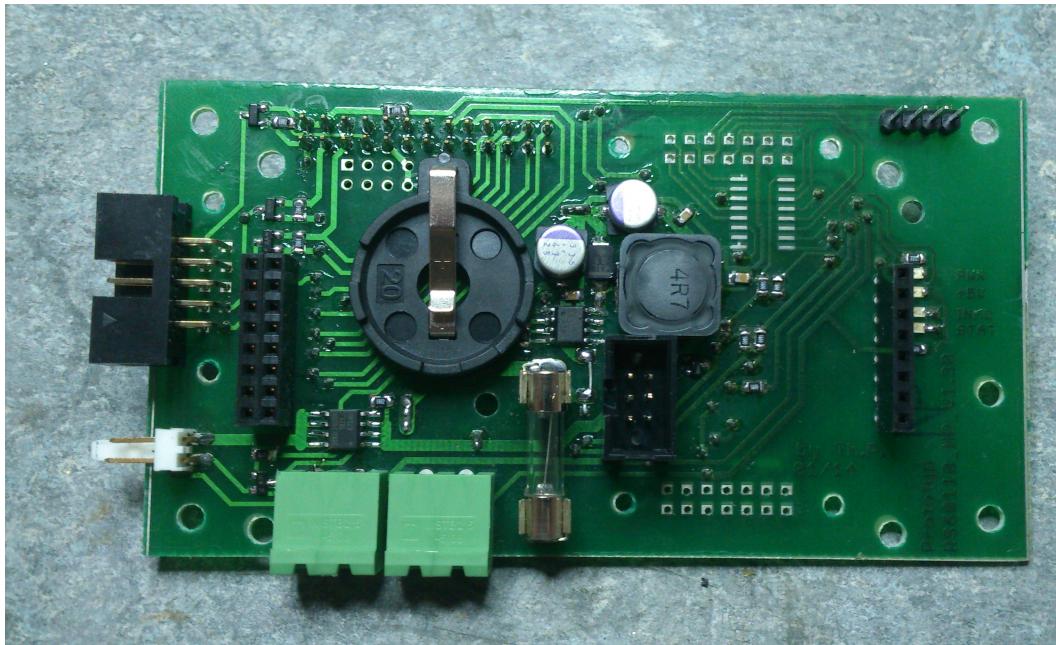


Das Board bekam bereits ein neues Layout, welches unter anderem Masseschleifen entfernte, und Status LEDs beinhaltet, die ein einfaches Debuggen ermöglichen sollen. Auch wurde der Schaltregler optimiert, welcher für das Board essentiell ist.

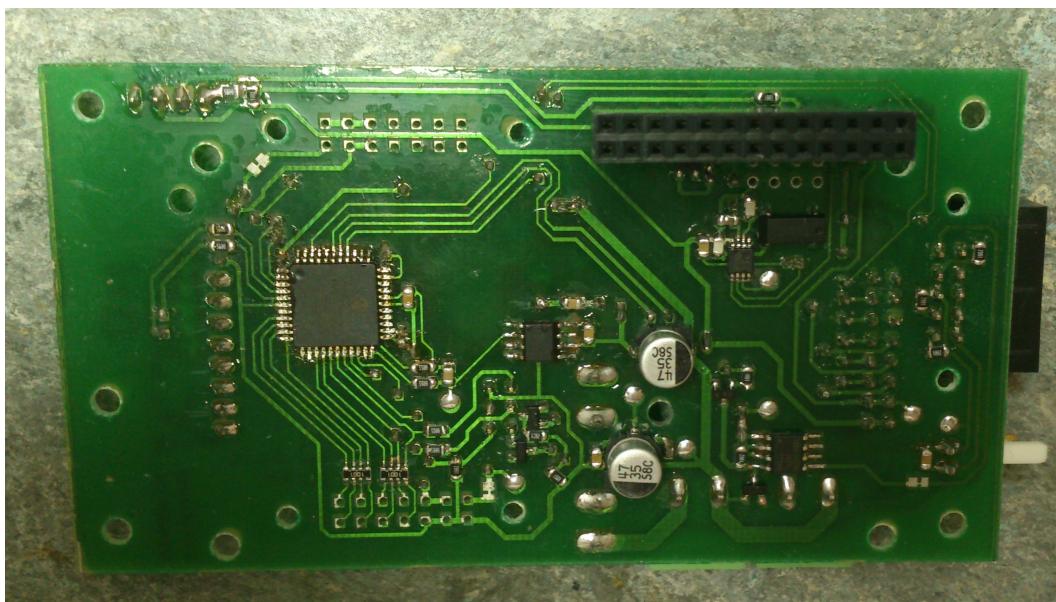


## Bestückung

### *Top-Layer*



### *Bottom-Layer*



## Inbetriebnahme der Hardware

### **Flashen des Steuer-PIC**

Es wird ein ICSP-Programmiergerät, welches einen PIC18F4420 flashen kann am board angeschlossen (ICSP-Stecker). Der PIC sollte korrekt erkannt werden, und kann dann mit der Firmware beschrieben werden.

### **Testen des Step-Down-Wandler**

1. Es wird am Eingang des Board ein Netzteil mit folgenden Einstellungen angeschlossen:  
 $I_{max} \approx 100mA$ ,  $U_a = 5V$
2. Am Spannungs-Ausgang für den Raspberry Pi wird ein Multimeter angeschlossen, welches eine Spannung von ungefähr 5V anzeigen sollte.
3. Jetzt wird die Spannung des Netzteiles langsam auf etwa 10-12V gesteigert, die Spannung am Multimeter darf dabei eine Spannung von 5,1-5,2V nicht überschreiten!

Falls die Spannung höher als etwa 5,2V steigen sollte funktioniert der Step-Down-Wandler nicht ordnungsgemäß. Der Step-Down-Wandler auf der Platine muss überprüft und wenn möglich repariert werden. Eine zu hohe Spannung am 5V-Spannungskreis könnte ansonsten zu einer Beschädigung diverser Bauteile auf dieser, und anderen angeschlossenen Platinen führen.

### **Fehlerbehebung**

Problem	Fehlerursache
Keine der Status-LED leuchtet auf	Board nicht korrekt angeschlossen
	Sicherung durchgebrannt
	PIC nicht/falsch geflasht
+5V, PWR-Led leuchtet nicht	Tiefentladeschutz aktiv (kurz aufleuchtende STAT-LED)
	MOSFET schaltet nicht durch

### **Testen der Hardware**

Der PIC wird mit der speziellen TEST-Firmware beschrieben (siehe Flashen des Steuer-PIC). Dann werden die beiden 14-Poligen Stiftleisten 1:1 miteinander verbunden. Sobald das Board mit Spannung versorgt wird startet die Testroutine. Solange die Testroutine aktiv ist blinken beide Status-LEDs schnell. Wenn diese dann erfolgreich war, Leuchtet *STAT* dauerhaft. Falls die Testroutine fängt *INFO* zu blinken an.

## Überblick über die Status-LEDs

### PWR

Die PWR-LED zeigt an, ob angeschlossenen Verbraucher die Eingangsspannung zur Verfügung gestellt haben.

### +5V

Diese LED leuchtet auf, wenn Verbrauchern die +5V vom StepDown zur Verfügung stehen.

Wenn die LED nicht leuchtet sind alle externen Verbraucher (auch der RaspberryPi) von der Spannungsversorgung getrennt. Dies ist bei ansprechen des Tiefentladeschutzes der Fall.

### INFO

### STAT

## Inbetriebnahme der Software

Hier ist die Konfiguration einen Raspberry Pi der 2. Revision beschrieben. Wenn ein Raspberry Pi der 1. Revision verwendet wird muss i2c-1 durch i2c-0 ersetzt werden! Das verwendete Distribution ist das oft verwende Wheezy, bei Arch und anderen Linux Derivaten kann die Konfiguration abweichen!

### Aktivieren von I2C und SPI

1. In der Datei `/etc/modprobe.d/raspi-blacklist.conf` werden die beiden Schnittstellen aktiviert, indem die beiden Zeilen wie folgt auskommentiert werden:

```
#blacklist spi-bcm2708  
#blacklist i2c-bcm2708
```

2. In der Datei `/etc/modules` werden dann folgende Module hinzugefügt:

```
snd-bcm2835  
i2c-bcm2708  
i2c-dev
```

### Aktivieren der I2C-Echtzeituhr

1. Hinzufügen der folgenden Zeile in der Datei `/etc/modules`:

```
i2c:mcp7941x
```

2. Es werden 2. Zeilen in der Datei `/etc/rc.local` hinzugefügt (vor dem Befehl „exit 0“):

```
echo mcp7941x 0x6f > /sys/class/i2c-dev/i2c-1/device/new_device  
hwclock -s
```

3. Jetzt kann man die fake-hwclock deaktivieren

```
sudo update-rc.d fake-hwclock remove
```

4. Jetzt wird der Raspberry Pi neu gestartet, und dann die aktuelle zeit wie folgt in UTC gesetzt (natürlich das Datum durch das aktuelle ersetzen):

```
sudo date -s "9 JAN 2014 12:00:00"
```

5. Beim Herunterfahren wird dann ab sofort die Zeit in die Echtzeituhr geschrieben, und beim nächsten Starten daraus ausgelesen.

## Aktivieren des LCD-Touch-Displays (MI0283QT-9A)

### Neue Anleitung

<http://lallafa.de/blog/2013/07/watterotts-new-rpi-shieldbridge/>

Muss noch getestet werden!

### Alte Anleitung

#### Herunterladen aller benötigten Dateien

- SD-Karten-Image (Wheezy)
- <http://www.lallafa.de/files/raspi/raspi-linux-3.6.11-ili9341-ads7846.tar.gz>

#### Image installieren

Als erstes wird das Wheezy Image ganz normal auf die SD-Karte gespielt. Wenn man den Raspberry Pi startet wird man vermutlich aber nichts außergewöhnliche bemerken.

#### vorkompilierte Kerneldateien kopieren

Wenn man ein Linux-System hat, dann steckt man die fertig beschriebene SD-Karte hinein, und bekommt dann 2 Partitionen angezeigt (Bei mir 59MB und 1,9GB). Bei Windows, naja, der kennt vermutlich nur eine. So sollte auch klar sein dass Windoof für diesen Schritt ungeeignet ist (zu mindestens ohne Spezialprogramme).

Die folgenden Schritte sollten mit dem Terminal ausgeführt werden, da zu mindestens bei mir Administrator-rechte benötigt wurden.

Wenn man im Terminal ist soll als erstes in das Verzeichnis des vor-kompilierten Kernels gewechselt werden. Da das ganze auf einzelne Partitionen abzielt, die bei alle einen anderen Namen haben ist dieser bei mir so geschrieben: <name der partition>. Dort soll die richtige Bezeichnung für die jeweilige Partition in /media eingetragen werden. Es sollte klar sein dass die betreffenden Partitionen vorher auch gemountet werden müssen (was bei mir automatisch geschehen ist).

#### boot-Ordner

In der kleineren Partition befinden sich normalerweise die boot-Dateien. Z.b. auch eine config.txt, mit der man den Raspberry pi übertakten,... kann. In diese Partition wird jetzt der Inhalt des Ordners des vor-kompilierten Kernels mit dem Namen boot hinein geschoben, und sollten ein paar Dateien ersetzen.

```
sudo cp -R boot/* /media/<name der boot partition>/
```

### ***root-Ordner***

Dann kommt der Inhalt vom Ordner *root* dran. Dieser kommt in die 2. Partition, in der sich die wohl bekannten linux-ordner befinden. Dabei kommt dieser Ordner nicht in den gleichnamigen Ordner */root/*, sondern wird direkt in dessen Wurzelverzeichnis geschoben, und erweitert so den Ordnerinhalt von */lib/*.

```
sudo cp -R root/* /media/<name der root partition>/
```

### ***Testen***

Man könnte gleich weitermachen, aber ich würde empfehlen die SD-Karte zu entmounten, und das ganze testen. Wenn es funktioniert, und das Display korrekt verbunden ist sollte schon ein Boot-Screen erscheinen, und man sich einloggen können. Wenn man aber *startx* eingibt wird man nur einen schwarzen Bildschirm sehen, und das wollen wir noch ändern. Früher traten mehrere Display Fehler auf, was aber jetzt behoben sein sollte.

### ***Installieren von xserver-xorg-video-fbdev***

Bei mir war dieser Schritt nicht notwendig, wenn es aber nicht installiert ist wird es einfach nachinstalliert:

```
sudo apt-get install xserver-xorg-video-fbdev
```

### ***X11-config-Datei erstellen***

Jetzt muss nur noch folgende Datei erstellt werden: */usr/share/X11/xorg.conf.d/99-fbdev.conf*

Dessen Inhalt sollte wie folgt sein:

```
Section "Device"
    Identifier "myfb"
    Driver "fbdev"
    Option "fbdev" "/dev/fb1"
EndSection
```

Jetzt sollte alles funktionieren, und wenn man *startx* eingibt, sollte auf dem display ein kleiner Desktop erscheinen.

## PIC-Firmware

### I<sup>2</sup>C

#### Schreiben

Das erste übertragene Byte setzt den Schreib/Lese-Index fest. Dieser wird dann nach jedem weiteren Schreib/Lesebefehl jeweils um 1 inkrementiert, solange er nicht neu gesetzt wurde.

Index	Aktion	Information
0x01	Setze TRISB	1... Eingang, 0... Ausgang
0x0F	Setze PORTB	1... High, 0... Low

#### Lesen

Index	Aktion	Information
0x01	Lese TRISB	1... Eingang, 0... Ausgang
0x0F	Lese PORTB	1... High, 0... Low
0x10	ADC - 0 VOLTAGE-HIGH-BYTE	
0x11	ADC - 0 VOLTAGE-LOW-BYTE	
0x12	ADC - 1 VOLTAGE-HIGH-BYTE	
0x13	ADC - 1 VOLTAGE-LOW-BYTE	
0x14	ADC - 2 VOLTAGE-HIGH-BYTE	
0x15	ADC - 2 VOLTAGE-LOW-BYTE	
0x16	ADC - 3 VOLTAGE-HIGH-BYTE	
0x17	ADC - 3 VOLTAGE-LOW-BYTE	
0x18	ADC - 4 VOLTAGE-HIGH-BYTE	
0x19	ADC - 4 VOLTAGE-LOW-BYTE	
0x1A	ADC - 5 VOLTAGE-HIGH-BYTE	
0x1B	ADC - 5 VOLTAGE-LOW-BYTE	
0x1C	ADC - 6 VOLTAGE-HIGH-BYTE	
0x1D	ADC - 6 VOLTAGE-LOW-BYTE	
0x1E	ADC - 7 VOLTAGE-HIGH-BYTE	
0x1F	ADC - 7 VOLTAGE-LOW-BYTE	
0x20	ADC - VCC VOLTAGE-HIGH-BYTE	
0x21	ADC - VCC VOLTAGE-LOW-BYTE	
0x22	ADC - +5V VOLTAGE-HIGH-BYTE	

<b>Index</b>	<b>Aktion</b>	<b>Information</b>
0x23	ADC - +5V VOLTAGE-LOW-BYTE	

asdfasdf