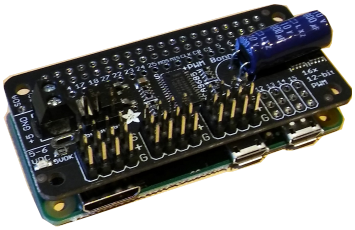

Empfänger RC Pi

RC Pi

Bernd Hinze

Software Developer im Unruhestand. Hobby in den 80er Jahren - Modellbau. Baut nun mit seinem Enkel Modellautos und Boote.

In den 80er Jahren musste man sehr viel selbst bauen. Das ist heute anders. Man benötigt bei entsprechendem Kleingeld eigentlich nur einen Kreuzschlitzschraubendreher und baut sich mit Fertigteilen das schönste Modell zusammen. Die Fernsteuerung kommt meist fast gratis dazu. Lernt man dabei etwas? Da ich auch früher Sender und Empfänger selbst entwickelte und baute, stellte ich mir die Frage, ob das nicht auch mit einem Raspberry Pi möglich ist. Das Resultat zum Nachbauen sehen sie hier.

„Es war ein tolles Gefühlt als das selbstgebaute Boot mit der RC Pi Fernsteuerung über den See jagte“

Schwierigkeitsgrad:
- Mittel

Benötigte Materialien
- eigener PC

Modell-Empfänger:
- Raspberry Pi Zero W
- 16-Channel, 12-bit PWM Board with PCA9685 (Adafruit)
- ADC mit ADS1115 (Option)

ca. 25 - 30 €

Sender:
- Smartphone

zusätzlich optional:
- Gamepad (USB)
- 5 V Powerbank rund
- Raspberry Pi Zero W

ca. 40 € für optionale Teile

Modell mit
- Servo (ADS-5 o.ä)
- Akku 7,2 V 2000mA/h
- ESC Controller mit 5 V BEC (Pulstec 45 A)

Bausatz ab 69 € incl. Servo, Motor, ESC

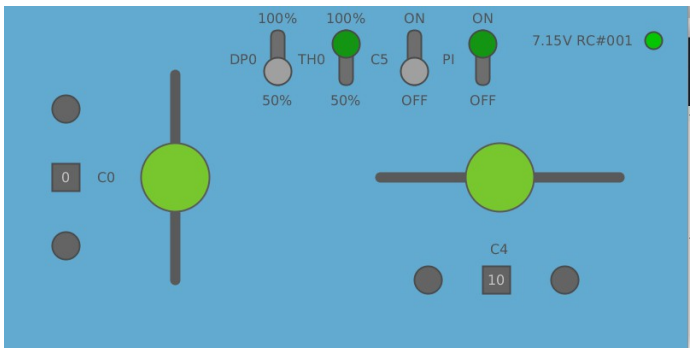

Bild 1: Senderapplikation für Smartphone

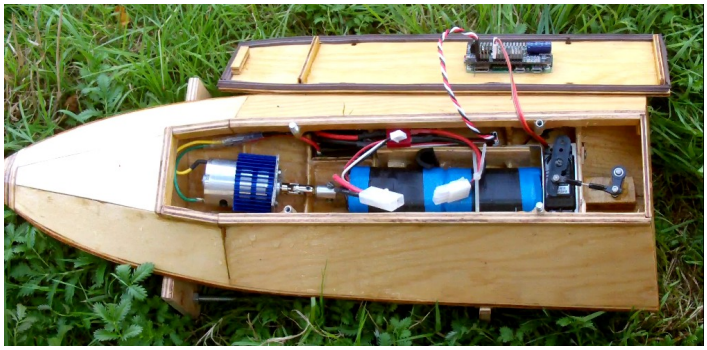

Bild 2: Rennboot aus Holz mit allen Einbauten

Bild 3: Optionales Gamepad mit Remote Screen und Raspberry Pi

Kontaktaufnahme

In einer ersten Trockenübung steuern sie mit der Smartphone App ein Servo in Echtzeit an. Dazu sind folgende Schritte erforderlich:

- Lokales Netz mit Hotspotfunktion des Smarphones einrichten
- Raspberry Empfänger aufbauen, Software installieren und konfigurieren
- App auf das Smartphone laden

01 Richten sie ein Hotspot auf dem Smartphone ein und erstellen sie mit den eingestellten Parametern eine Textdatei mit dem Dateinamen `,wpa_supplicant.conf‘`.

02 Die SD-Card des Raspberry Pi laden sie mit dem neuesten Lite Image.
https://www.raspberrypi.org/downloads/raspbian/
Nutzbare Tools sind unter Linux ‚balenaEtcher‘ und unter WINDOWS ‚Win 32 Disk Imager‘.
Noch bevor die SD-Card im Raspberry Pi genutzt wird, sollten sie zwei Dateien, die o.g. `,wpa_supplicant.conf‘` und eine leere Datei mit dem Namen `,ssh‘` auf die ‚Boot‘ Partition der SD-Card kopieren. Das PWM-Bonnet Board stecken sie auf den Raspberry Pi. Vorher bitte die im Bild 4 dargestellte Brücke auf das PWM-Bonnet Board löten. So erhält der Raspberry Pi eine Spannung von 5V vom PWM Board.
Mit `,ssh -l pi [IP]‘` am Raspberry Pi einloggen. Der PC muss ebenfalls am Hotspot des Smartphones angemeldet sein. Die genutzten IP-Adressen ermitteln sie mit folgendem Befehl:
`nmap -sP 192.168.43.0/24` (Beispiel)
Nach der SW-Aktualisierung führen sie `,sudo raspi-config‘` aus und aktivieren das I2C Interface. Nicht vergessen das ‚pi‘ Passwort ebenfalls zu ändern.

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
network={
  ssid="[SSID]"
  psk="[Passwort]"
  id_str="netzwerk_a"
}
```

`,wpa_supplicant.conf‘`

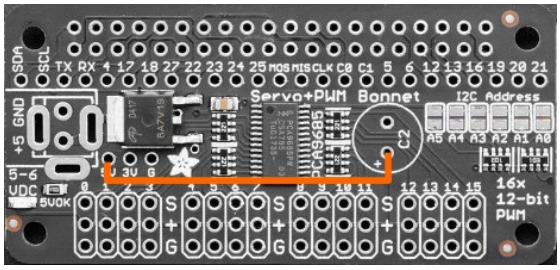

Bild 4: Brücke auf dem PWM Board

Software des Raspberry update/einrichten

```
sudo apt-get update
sudo rpi-update
sudo raspi-config
```

Immer gut 'Midnight Commander‘

```
sudo apt-get install mc
```

SMBus und I2C Tools instalieren

```
sudo apt-get install python3-pip
sudo pip3 install netifaces
sudo apt-get install python3-smbus

sudo apt-get install i2c-tools
```

Bild 5: Kommandos zu Installation

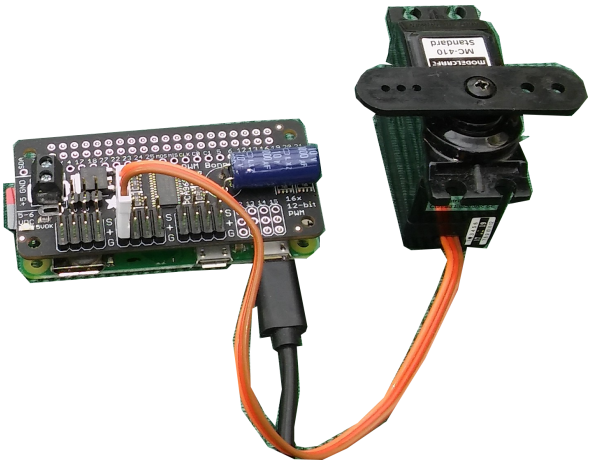

Bild 6: Servoansteuerung PiRx

Mit dem Kommando

`> sudo i2cdetect -y 1` testen sie, ob das PWM Board erkannt wurde. Es wird eine Tabelle mit alle erkannten I2C Geräte ausgegeben.

```
pi@RemoteControl:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- --
```

Nachdem sie sich eingeloggt haben, können sie mit ‚sftp' des ‚mc' commanders oder einem anderen Tool die folgenden Dateien

- **ads1115.py | ADS1115 Treiber**
- **pca9685.py | PWM Board Treiber**
- **rcapp.py | Empfänger**
- **rccfg.py | Konfiguration**

in ein Verzeichnis des Raspberry Pi unterhalb **home/pi** übertragen.

Für ihr Modell muss nur die Datei `‚rccfg.py'` angepasst werden.

Die Applikation kann automatisch mit dem ‚systemd' Dienst nach dem Booten gestartet werden. Dazu müssen sie eine Datei mit dem Namen `‚myrc.service'` im Systemverzeichnis `‚/lib/systemd/system'` anlegen und mit dem Kommando `‚chmod'` ausführbar machen.

Die Datei hat den folgenden Inhalt:

```
[Unit]
Description=myrc

[Service]
ExecStart=/home/pi/prj/rcapp.py
TimeoutSec=3
StandardOutput=null

[Install]
WantedBy=multi-user.target
Alias=myrc.service
```

Den Dienst aktivieren sie mit den Kommandos

```
> sudo systemctl enable myrc.service
> systemctl daemon-reload
```

Nach dem Booten startet die Applikation automatisch. Um zu verhindern, dass Fehler zu einem Abbruch dieses automatischen Aufrufs führen, sollte die komplette Appplikation einmal mit

`> python3 rcapp.py`

gestartet werden. Anschließend mit Ctrl C abbrechen.

03 Smartphone App installieren

Die Senderapplikation für das Smartphone ist als vorcompilierte Anwendung ‚phonetx_release_signed.apk' verfügbar und kann nach dem Download auf das Handy und temporärer Freigabe fremder Quellen installiert werden. Damit sind alle Voraussetzungen vorhanden das Servo mit dem Smartphone zu steuern.

- Hotspot im Smartphone einschalten,
- Daten ausschalten
- PhoneTx starten
- Raspberry Py mit der Empfänger-App booten

Wenn alles richtig konfiguriert wurde, wechselt der Kommunikationsindikator in der rechten oberen Ecke von Rot nach Grün. Nun kann der Servo auf Kanal 3 gesteuert werden.

Die Smartphone App wurde mit 'Processing' - einem JAVA Framework entwickelt und kann an eigenen Bedürfnissen angepasst werden.

Aus meiner Sicht lässt sich ein Boot gut mit der Smartphone-App steuern. Schneller reagierende Modelle, wie Rennautos aber nur mit viel Übung.

Deshalb wurde zusätzlich eine Applikation 'GamepadTx' entwickelt, die Kommandos eines Gamepads verarbeitet und via WLAN kompatible Steuerdaten an den PiRx sendet.

Da ein Gamepad in der Regel kein eigenes Display zur Statusanzeige besitzt, werden diese optional auf einem Remote Screen angezeigt.

'GPScreen' ist eine Smartphone-App, die Daten vom Gamepad empfängt und anzeigt.

Gamepad Spielereien

Bild 7: Gamepad Transmitter

Es sind zunächst ein paar mechanische Arbeiten notwendig.

01 Kabel

Ersetze sie das lange USB Kabel des Gamepads durch ein kurzes mit Micro USB-B Stecker. Auf die Farben sollte man sich nicht immer verlassen. Bei mir waren die Drähte der Datenleitungen vertauscht. Standard sind folgende Zuordnungen:

+ VCC - Rot
+ D - Gelb
- D - Grün
GND - Schwarz

Ein Test am Raspberri Pi ist mit folgendem Befehl möglich:

```
> ls /dev/input
```

Der Output muss sich von dem ohne angestecktem Gamepad unterscheiden.

02 Der Handy Halter

(hier X-Box 360) muss ein wenig angepasst werden. Der Durchbruch für das USB Kabel ist nach unten zu verlängern. Zusätzlich ist ein PVC- Streifen auf der Frontseite des Gamepads aufzukleben, um den Halter an das Gamepad anzupassen.

Auf der Rückseite des Schmartphone Halters wird ein PVC-Winkel angeklebt, an dem der Raspberry mit einem Kabelbinder befestigt wird.

Es sind noch ein paar Aussparungen in den Winkel zu feilen, um die USB Buchsen benutzen zu können.

Die Powerbank wird ebenfalls mit Kabelbindern am Schmartphone Halter befestigt.

03 Softwareinstallation

Jedem Bedienelement eines Gamepads ist ein sogenanntes ‚Event' zugeordnet. Damit muss man sich zunächst vertraut machen, um anschließend die Konfiguration durchführen zu können.

Ein paar Zeilen Python Code reichen.

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-

from evdev import InputDevice
gamepad = \
InputDevice('/dev/input/event22')

for event in gamepad.read_loop():
    print event.code
```

Die Eventnummer in ‚/dev/input/event..' ist abhängig von der PC- oder Rpi Konfiguration. man findet sie, indem man

```
> python3 GPcfg.py
```

eingibt, ohne das Gamepad anzustecken. Nach Aufforderung steckt man das Gamepad an und erhält die ‚InputDevice' Nummer.

Konfiguration

Prinzipiell lassen sich alle Programme auch zum Test auf einem PC starten.
Dazu sind die Parameter wie folgt zu setzen:

Datei	RPI	PC
rccfg.py	SIM = False	SIM = True
rccfg.py	ADS = False oder True	ADS = False
rccfg.py	ifname = ‚wlan0‘	ifname = ‘...‘

Tabelle 1: Target Konfiguration

Die eigentliche Konfiguration des Empfängers ist in rccfg.py kommentiert und bedarf keiner weiteren Erläuterung. Das optionale ADS Board wird ebenfalls über den I2C-Bus angesteuert.

Unterschiede zu TelDaControl

Dieses Gesamtsystem ist eine Überarbeitung und Weiterentwicklung der Software ‚TelDaControl‘ [1]. Die wesentlichen Änderungen sind:

- Beseitigung des objektorientierten Overheads und Reduzierung der Klassen auf das absolut notwendigste Mass.
- Zur Vermeidung vom ‘GIL’ (Global Interpreter Lock’ ca. 5 ms) bei der Kommunikation mit ’Threads‘ werden ’Queues’ verwendet.
- Änderung der Telegrammkodierung. Gegenüber TelDaControl und RcPi-Release 1, wird eine einheitliche Telegrammstruktur für alle zu übertragenden Daten verwendet (siehe Anhang).
- Optionale Verwendung eines I-Gliedes (Integrator) für den Antriebskanal. Damit kann eine Absenkung der Pi-Versorgungsspannung unter 5V durch zu hohe Stromänderung des Antriebes vermieden werden (z.B Start mit voller Auslenkung des Steuerhebels).
- Verbessertes Startup Verhalten

Folgende Software steht zur Verfügung:

- PhoneTx - Android App
- RCPC - wie PhoneTx aber für PC
- PiRx - Empfänger App für Raspberry Pi
- GamepadTx - Rpi Sender für Gamepad
- GPScreen - Android Remote Screeen

Sie werden sich fragen, wie groß ist denn die Reichweite. Ist das mit Wlan überhaupt ausreichend? Ich habe einige Reichweitentests mit unterschiedlichen Konfigurationen gemacht. Die Resultate sehen sie in der folgenden Tabelle:

Empfänger	Access-point	Sender	Reichweite
Rpi-Zero WH	Handy	Handy od. Gamepad	25 m
Rpi-Zero WH	WLan-Router mit externer Antenne	Handy od. Gamepad	100 m
Rpi-Zero mit externem USB Stick (plus externer Antenne)	WLan-Router mit externer Antenne	Handy od. Gamepad	ca. 200 - 400 m

Tabelle 2: Reichweite

Startverhalten

Wie im RC Sport üblich, sollte beim Einschalten folgende Reihenfolge eingehalten werden:

1. Hotspot (Smartphone oder externer AP)
2. Remote Screen falls vorhanden
3. PhoneTx oder GamepadTx
4. PiRx

Es sollte immer gewartet werden, bis die jeweilige Komponente hochgefahren ist.

Abkürzungen

ADS	Analog Digital Konverter mit ADS1115
BEC	Battery Elimination Circuit
I2C	Serial 2 Wired Interface
SFTP	SSH File Transfer Protocol
SIM	Simulation
UDP	User Datagram Protocol

Referenzen

[1]	https://github.com/monbera/TelDaControl
[2]	https://processing.org

Anhang - Telegramm Definition und Startup Verhalten

Für die Kommunikation zwischen Sender und Empfänger wird ein einfaches zyklisches UDP-Telegramm verwendet.

Für die Übertragung über WiFi werden die n Bytes in Halbbytes zerlegt und dadurch auf 2 * n Telegrammbytes aufgeteilt. (Im höherwertigen Halbbyte ist immer eine 3 eingetragen)

Wertebereich: 30H ... 3FH.

Außerdem werden folgende Steuerzeichen hinzugefügt:

STX (02H) Startkennzeichen vorangestellt

CR (0DH) Endekennzeichen angehängt

Nettodaten:

00 FF 00 00 00 00 00 00 8E A5

Zeichenstrings auf der Schnittstelle:

02 30 30 3F 3F 30 30 30 30 30 30 30 30 30 30 30 38 3E 3A 35 0D

Telegrammarten

Die Struktur der Telegramme wurde vereinheitlicht und enthält im ersten Byte nun eine eindeutige Telegramm ID.

RC_BC : Empfänger macht die eigene IP bekannt und überträgt Sensordaten
Tx_Ctr : Steuertelegramm für Servos und andere Aktoren
Tx_BC : Der Sender macht die eigene IP bekannt
Tx_RSC : Ein optionales Gamepad mit Raspberry Pi als Sender übermittelt an einen Remote Screen (RSC) interne Daten zur Visualisierung
RSC_BC : Lebenszeichen des RSC mit Telegramm ID

Name	Receiver		Transmitter		Remote Screen	ID
Rx_BC	x	-->	x			1
Tx_Ctr	x	<--	x			2
Tx_BC			x	-->	x	3
Tx_RSC			x	-->	x	4
RSC_BC			x	<	x	5

Tabelle 3: Telegramme

Rx_BC

Byte		Value/ Range	
0	ID	**1**	Telegramm ID
1	IP.1	0..255	IP.1 des Empfängers
2	IP.2	0..255	IP.2 des Empfängers
3	IP.3	0..255	IP.3 des Empfängers
4	IP.4	0..255	IP.4 des Empfängers
5	S1.1	0..99	Sensor Daten, Ganze Zahl
6	S1.2	0..99	Sensor Daten, Nachkommawert

Tabelle 4: Telegramm Rx -> TX (Broadcast)

Tx_Ctr

Byte		Value/ Range	
0	ID	**2**	Telegramm ID
	n = 0 .. 32		
1 + 3 * n	HDR	0..255	Header 255 = Control Header 127 = Trimm
2 + 3 * n	CHA	0..15	Kanalnummer
3 + 3 * n	VAL	0..254	Control Value, Center = 127

Tabelle 5: Telegramm Tx -> Rx

Tx_BC

Byte		Value/ Range	
0	ID	**3**	Telegramm ID
1	IP.1	0..255	IP.1 des Senders
2	IP.2	0..255	IP.2 des Senders
3	IP.3	0..255	IP.3 des Senders
4	IP.4	0..255	IP.4 des Senders

Tabelle 6: Telegramm Tx_BC

Tx_RSC

Byte		Value/ Range	
0	ID	**4**	Telegramm ID
1	IP.1	0..255	IP.1 des Empfängers
2	IP.2	0..255	IP.2 des Empfängers
3	IP.3	0..255	IP.3 des Empfängers
4	IP.4	0..255	IP.4 des Empfängers
5	STA	0..3	Kommunikationsstatus 0 = default 1 = connect 2 = lost
6	S1.1	0..99	ganze Zahl
7	S2.2	0..99	Nachkommawert
	n -1 times		START = 8
START + 4 * n	CH	0..15	Channel
START + 1 + 4 * n	DRn	0..100	Dual Rate
START + 2 + 4 * n	TRn	0..50	Trimm Rate
START + 3 + 4 * n	VALn	0..254	Control Value

Tabelle 7: Telegramm Tx_RSC (nur bei Verwendung des Gamepads)

RSC_BC

Byte		Value/ Range	
0	ID	**5**	Telegramm ID

Tabelle 8: Telegramm RSC_BC (nur bei Verwendung des Gamepads)

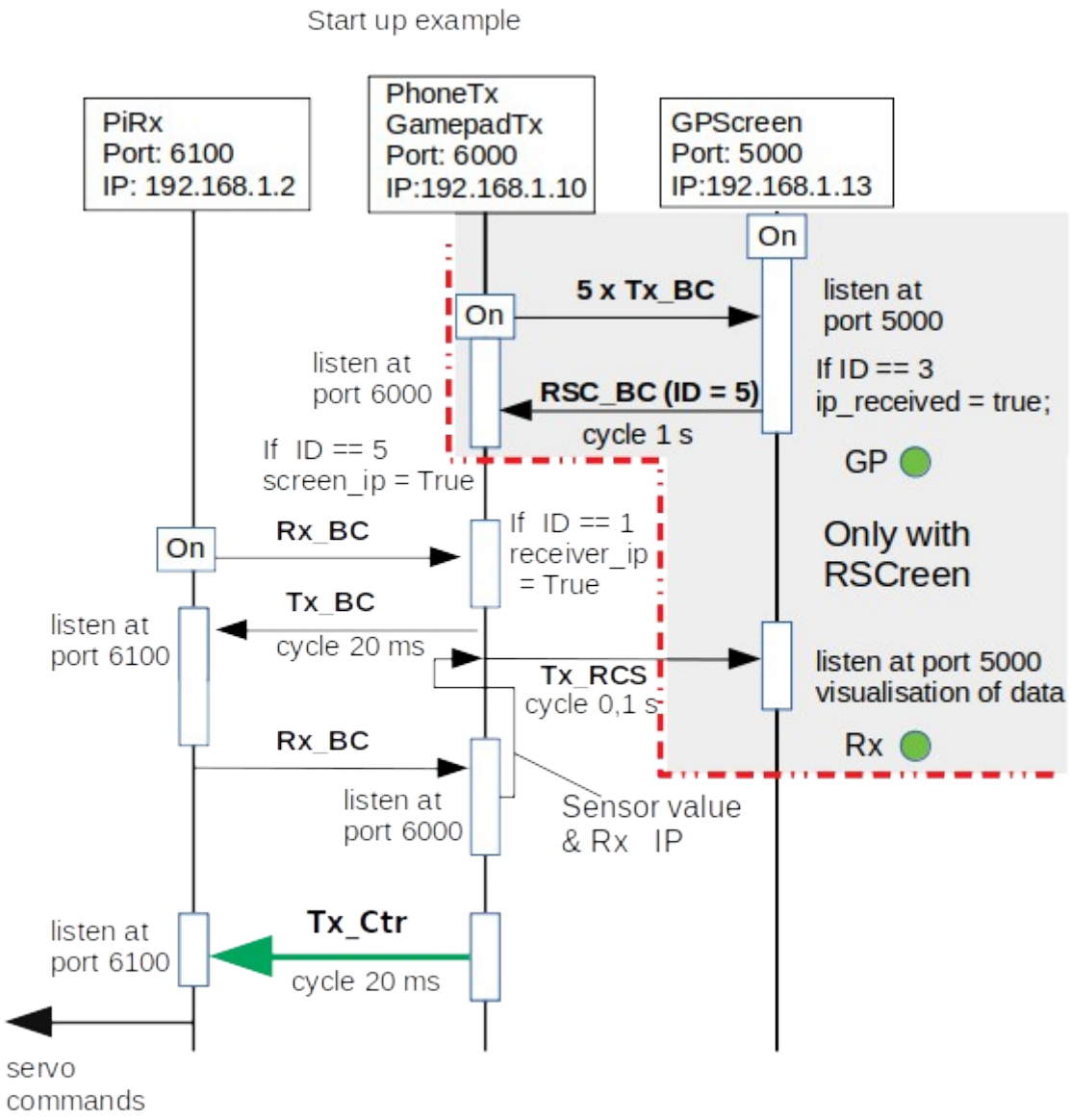

Bild 8: Dynamisches Start Up Verhalten

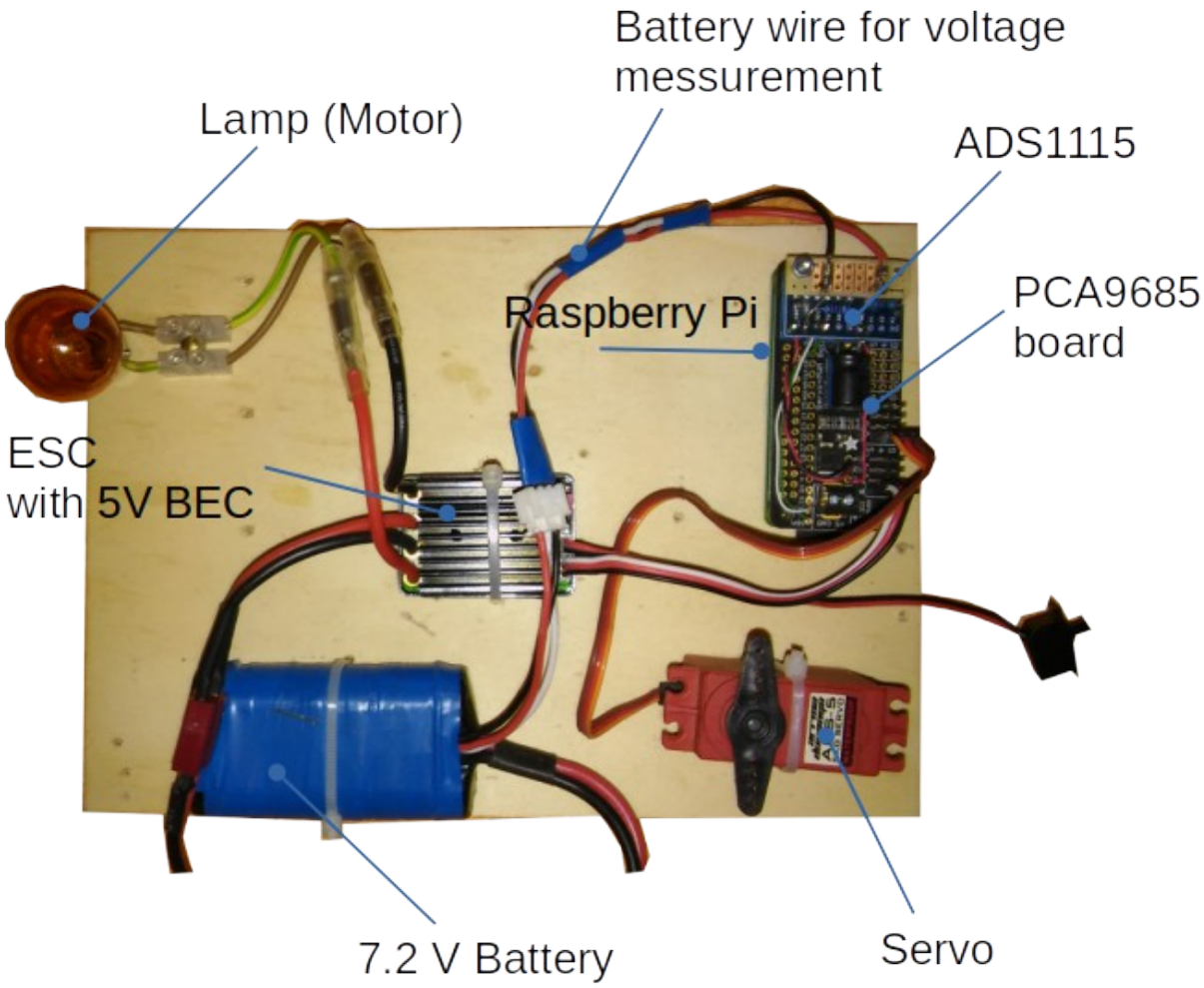

Bild 9: Testplattform

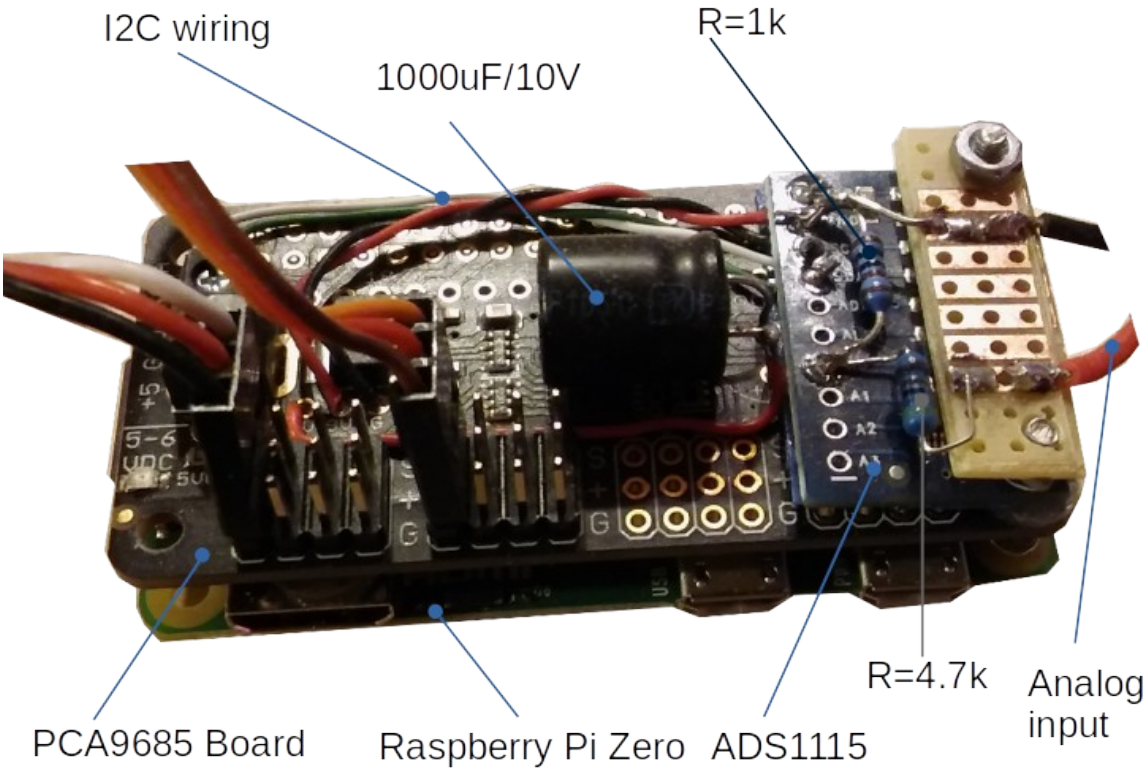

Bild 10: Empfänger mit analog Eingang ADS1115

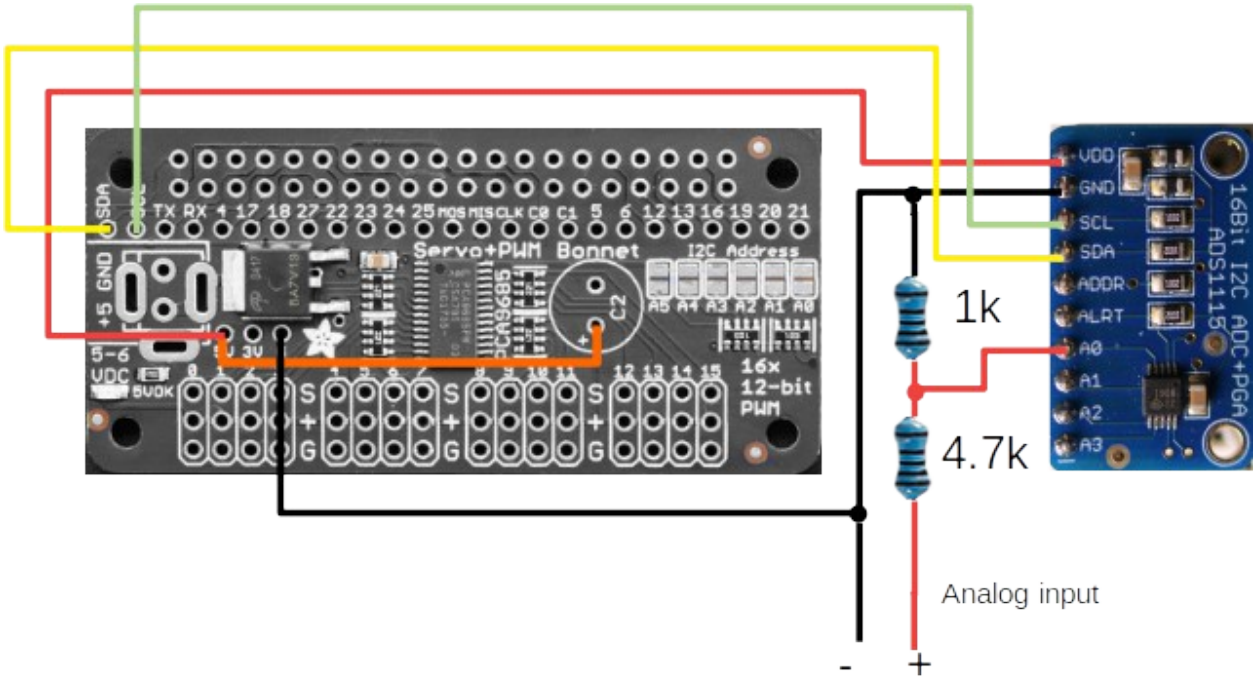

Bild 11: Kopplung PCA9685 Board - ADS1115 Board