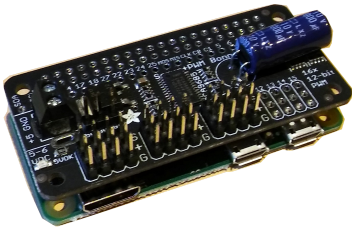

Receiver RC Pi

RC Pi

Bernd Hinze

Software developer in retirement. Hobby in the 80s - model making. Now builds model cars and boats with his grandson.

In the 80s, you had to build a lot yourself. This is different today. You only need a Phillips screwdriver for the right change and you can build the most beautiful model with prefabricated parts. The remote control is usually added almost for free. Do you learn anything? Since I used to develop and build transmitters and receivers myself, I asked myself if this is possible with a Raspberry Pi. Here you can see the result of my work.

„It was a great feeling when the self-built boat with the RC Pi remote control chased across the lake“

Difficulty level:
- midle

Required materials
- own PC

Receiver for model:
- Raspberry Pi Zero W
- 16-Channel, 12-bit PWM Board with PCA9685 (Adafruit)
- ADC with ADS1115 (option)

approximately 25 - 30 €

Transmitter:
- Smart phone

optional in addition:
- Game pad (USB)
- 5 V Power bank round
- Raspberry Pi Zero W

approximately. 40 € for optional parts

Model with
- Servo (ADS-5 o.ä)
- Battery 7,2 V 2000mA/h
- ESC controller with 5 V BEC (Pulstec 45 A)

Kits from 69 € inc. servo, motor and ESC.

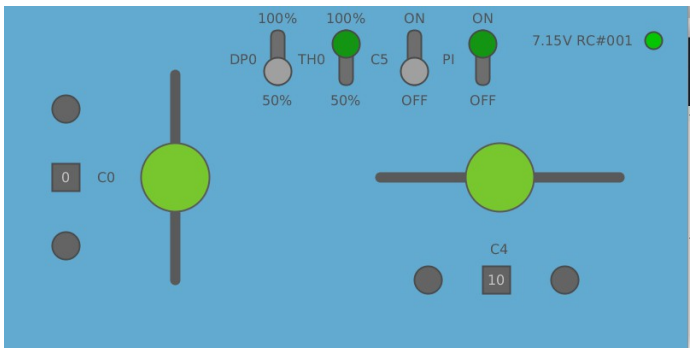

Figure 1: Transmitter Application for Smartphone

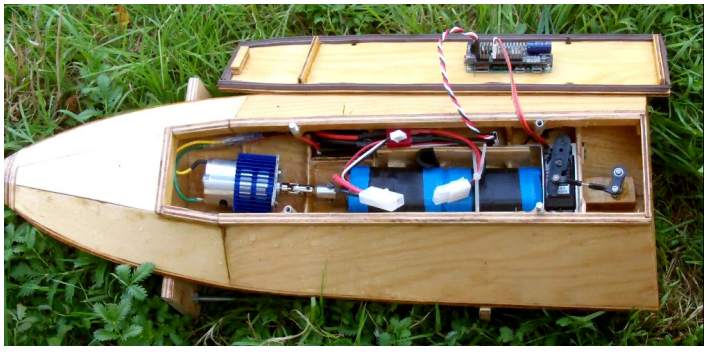

Figure 2: Wooden Speedboat with all Parts

Figure 3: Optional Gamepad with Remote Screen and Raspberry Pi

First Contact

In a first dry run you control a servo in real time with the smart phone app. To do this, you need to take a few simple steps:

- Set up a local network with the smart phone's hot spot function
- Set up the Raspberry receiver, install and configure the software
- Downloading the app to your smart phone

01 Set up a hot spot on the smart phone and create a text file with the setup parameters `‚wpa_supplicant.conf‘`.

02 Load the SD card of the Raspberry Pi with the latest ‚Lite Image‘.
https://www.raspberrypi.org/downloads/raspbian/
Usable tools are under Linux 'balenaEtcher' and under WINDOWS 'Win 32 Disk Imager'. Before using the SD card in Raspberry Pi, you should copy two files, the above mentioned `'wpa_supplicant.conf'` and an empty file named 'ssh' to the 'boot' partition of the SD card. Put the PWM-Bonnet board on the Raspberry Pi. Please solder the bridge shown in Figure 4 to the PWM-Bonnet board first. This way the Raspberry Pi use a voltage of 5V from the PWM board.
Log in with `'ssh -l pi [IP]'` on the Raspberry Pi. The PC must also be logged on to the hot spot of the smart phone. The IP addresses used can be determined with the following command:
`nmap -sP 192.168.43.0/24` (example)
After the SW update, they execute ‘sudo raspi-config' and activate the I2C interface. Do not forget to change 'pi' password as well.

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

`‚wpa_supplicant.conf‘`

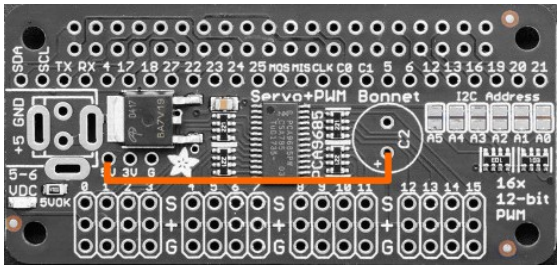

Figure 4: Short Cut on the PWM Board

Installing software of the Raspberry Pi

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

always good 'Midnight Commander‘

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

SMBus and I2C Tools installation

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

sudo apt-get install i2c-tools
```

Figure 5: Commands for Installation

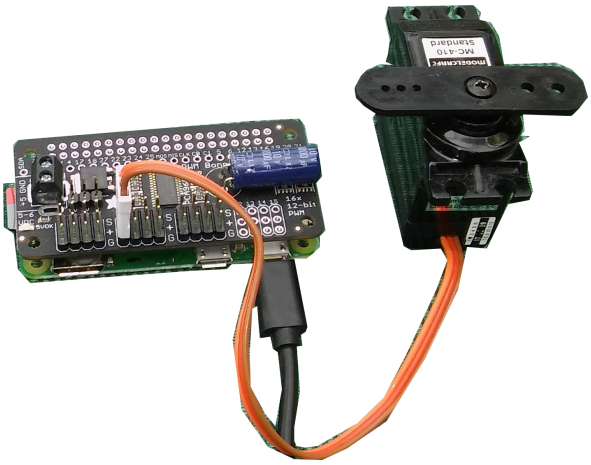

Figure 6: Servo Control PiRx

With the command

`> sudo i2cdetect -y 1` you can test whether the PWM board has been detected. A table with all detected I2C devices is displayed.

```
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 -- -- -- -- -- -- --
```

After you have logged in, you can use 'sftp' of the 'mc' commander or another tool to upload the following files

- **ads1115.py | ADS1115 library**
- **pca9685.py | PWM Board library**
- **rcapp.py | Receiver**
- **rccfg.py | Configuration**

in a folder of the Raspberry Pi below **home/pi**.

For your model only the file `'rccfg.py'` needs to be adapted.

The application can be started automatically with the 'systemd' service after booting. For this you have to create a file named `'myrc.service'` in the system directory `'/lib/system/system'` and make it executable with the command `'chmod'`.

The file has the following contents:

```
[Unit]
Description=myrc

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

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

You activate the service with the commands:

`> sudo systemctl enable myrc.service`

`> systemctl daemon reload`

After booting the application starts automatically. To prevent errors from cancelling this automatic call, you should start the entire application once with

`> python3 rcapp.py`

and check if all is running, then abort it with Ctrl. C.

03 Smart phone App Installation

The transmitter application for the smart phone is available as a precompiled application `'phonetx_release_signed.apk'` and can be installed after downloading it to the phone and temporarily releasing foreign sources. This means that all the requirements are met to control the servo with the smart phone.

- Switch on the hot spot on the smart phone
- Switch off data
- Start ‘PhoneTx’
- Boot Raspberry Py with the receiver app

If everything is configured correctly, the communication indicator in the upper right corner will change from red to green. Now the servo can be controlled on channel 3.

The smart phone app was developed with 'Processing' - a JAVA framework and can be adapted to your own needs.

From my point of view a boat can be controlled well with the smart phone app. Faster reacting models, like racing cars but only with a lot of practice.

Therefore an additional application 'GamepadTx' was developed, which processes commands of the game pad and sends control data compatible to the PiRx via Wlan. Since a game pad usually does not have its own display for status indication, these are optionally displayed on a remote screen. ‘GPScreen' is a smart phone app that receives and displays data from the game pad.

Gamepad Gimmicks

Figure 7: Game pad Transmitter

At first some mechanical work is necessary.

01 Cable

Replace the long USB cable of the game pad with a short one with a micro USB-B connector. You should not always rely on the colours. With me the wires of the data lines were interchanged. Standard are the following assignments: .

+ VCC - red
+ D - yellow
- D - green
GND - black

A test on the Raspberry Pi is possible with the following command:

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

The output must be different from the output without the game pad plugged in.

02 The Smartphone Holder

(here X-Box 360) has to be adjusted a little bit. The hole for the USB cable must be extended downwards. In addition, PVC strips must be glued to the front of the game pad to adapt the holder to the game pad.

On the back of the smart phone holder a PVC angle is glued on, to which the Raspberry is attached with a cable tie.

You have to file some cut outs into the angle to be able to use the USB sockets.

The power bank is also attached to the smart phone holder with cable ties.

03 Installation of Software

Each control element of a game pad has a so-called 'event' assigned. You must first familiarise yourself with this in order to be able to carry out the configuration afterwards.

A few lines of Python code are enough to catch the event for each button.

```
#! /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
```

The event number in '/dev/input/event...' depends on the PC or Rpi configuration.

you can find it by typing

```
> python3 GPcfg.py
```

without plugging in the game pad. When prompted you plug in the game pad and get the input device number.

Configuration

In principle, all programmes can also be started for testing on a PC.

To do this, the parameters must be set as follows:

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

Table 1: Target Configuration

The actual configuration of the receiver model is commented in rccfg.py and needs no further explanation.

Differences to TelDaControl

This complete system is a revision and further development of the software 'TelDaControl' [1].

The main changes are:

- Elimination of the object-oriented overhead and reduction of classes to the absolute minimum.
- To avoid the 'GIL' (Global Interpreter Lock, approx. 5 ms) when communicating with threads. Queues are used.
- Change of the telegram coding. Compared to ‘TelDaControl’ and RcPi Release 1, a uniform telegram structure is used for all data to be transmitted (See Annex).
- Optional use of an I-element (integrator) for the drive channel. This prevents the Pi supply voltage from dropping below 5V due to excessive current change of the drive (e.g. start with full deflection of the control lever).
- Improvement of start-up behaviour

The following software is available:

- PhoneTx - Android App
- RCPC - like PhoneTx but for PC
- PiRx - Receiver App für Raspberry Pi
- GamepadTx - Rpi Transmitter for Gamepad
- GPScreen - Android Remote Screen

You're going to wonder how big the range is. Is it enough with Wlan? I've done some range tests with different configurations.

The results are outlined in the following table:

Receiver	Acces-spoint	Transmitter	Range
Rpi-Zero WH	Handy	Handy or Gamepad	25 m
Rpi-Zero WH	WLan-Router with external antenna	Handy or Gamepad	100 m
Rpi-Zero with external USB Stick (plus external antenna)	WLan-Router with external antenna	Handy or Gamepad	approx. 200 - 400 m

Table 2: Range of Controlling

Start Up Behavior

As usual in RC sports, the following sequence must be observed when switching on:

1. Hot spot (Smart phone or external access point)
2. remote screen if available
3. PhoneTx or GamepadTx
4. PiRx

You should always wait until the respective component has started up.

Abbreviations

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

References

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

Annex - Telegram definition and startup behavior

A simple cyclic UDP telegram is used for the communication between transmitter and receiver.
For transmission via WiFi, the n bytes are divided into half bytes and thus split into 2 * n telegram bytes. (In the more significant half byte always a 3 is entered)
Value range: 30H ... 3FH.
The following control characters are also added:

STX (02H) Start code prefixed

CR (0DH) End indicator appended

Netto data:
00 FF 00 00 00 00 00 00 8E A5
Character strings on the interface:
02 30 30 3F 3F 30 30 30 30 30 30 30 30 30 30 30 38 3E 3A 35 0D

Telegram types

The structure of the telegrams has been standardised and now contains in the first byte a unique Telegram ID.

RC_BC : Receiver makes its own IP known and transmits sensor data
Tx_Ctr : Control telegram for servos and other actuators
Tx_BC : The sender makes his own IP known
Tx_RSC : An optical game pad as a transmitter transmitted to a remote screen (RSC) internal data for visualisation
RSC_BC : Life sign of the RSC with telegram 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

Table 3: Telegrams

Rx_BC

Byte		Value/ Range	
0	ID	**1**	Telegram ID
1	IP.1	0..255	IP.1 of the receiver
2	IP.2	0..255	IP.2 of the receiver
3	IP.3	0..255	IP.3 of the receiver
4	IP.4	0..255	IP.4 of the receiver
5	S1.1	0..99	Sensor data, integer number
6	S1.2	0..99	Sensor data, decimal point value

Table 4: Telegram Rx -> Tx (Broadcast)

Tx_Ctr

Byte		Value/ Range	
0	ID	**2**	Telegram ID
	n = 0 .. 32		
1 + 3 * n	HDR	0..255	Header 255 = Control values Header 127 = Trimm values
2 + 3 * n	CHA	0..15	Channel number
3 + 3 * n	VAL	0..254	Control value, center = 127

Table 5: Telegram Tx -> Rx

Tx_BC

Byte		Value/ Range	
0	ID	**3**	Telegramm ID
1	IP.1	0..255	IP.1 of transmitter
2	IP.2	0..255	IP.2 of transmitter
3	IP.3	0..255	IP.3 of transmitter
4	IP.4	0..255	IP.4 of transmitter

Table 6: Telegram Tx_BC

Tx_RSC

Byte		Value/ Range	
0	ID	**4**	Telegram ID
1	IP.1	0..255	IP.1 of receiver
2	IP.2	0..255	IP.2 of receiver
3	IP.3	0..255	IP.3 of receiver
4	IP.4	0..255	IP.4 of receiver
5	STA	0..3	Status of communication 0 = default 1 = connect 2 = lost
6	S1.1	0..99	Sensor data, integer number
7	S2.2	0..99	Sensor data, decimal point value
	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

Table 7: Telegram Tx_RSC (only with Gamepad)

RSC_BC

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

Table 8: Telegram RSC_BC (only with game pad)

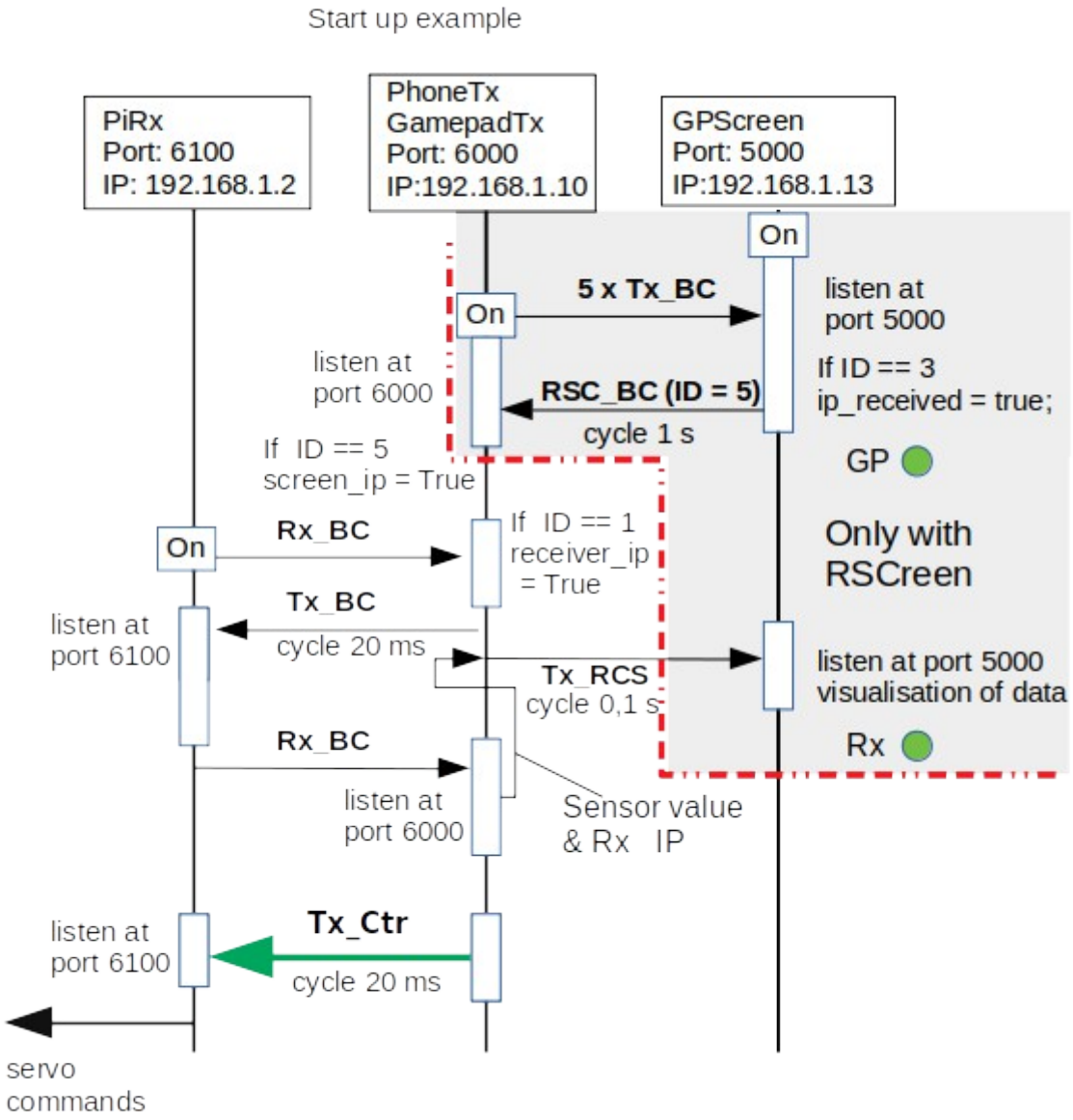

Figure 8: Dynamical Start Up Behavior

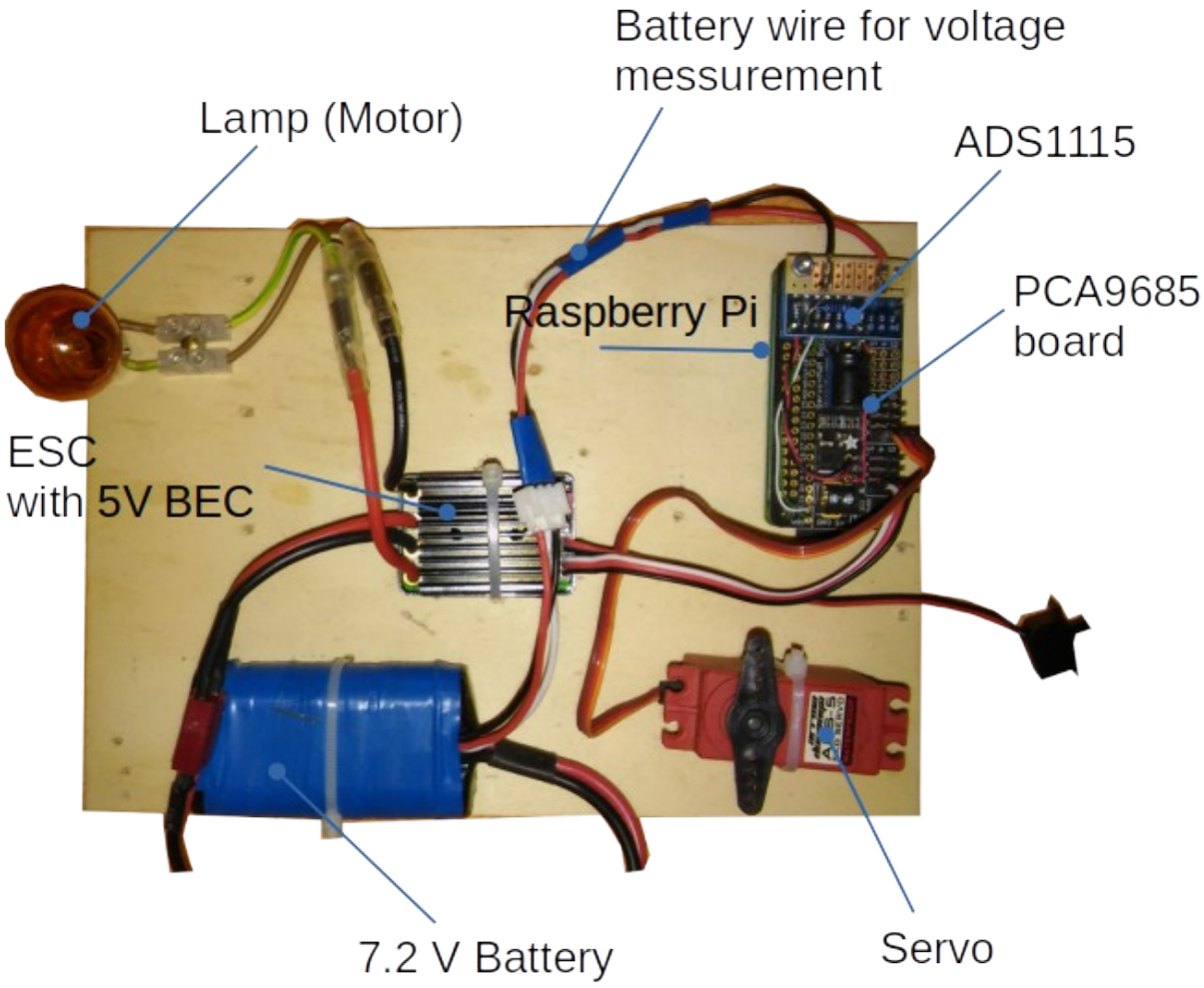

Figure 9: Testbed

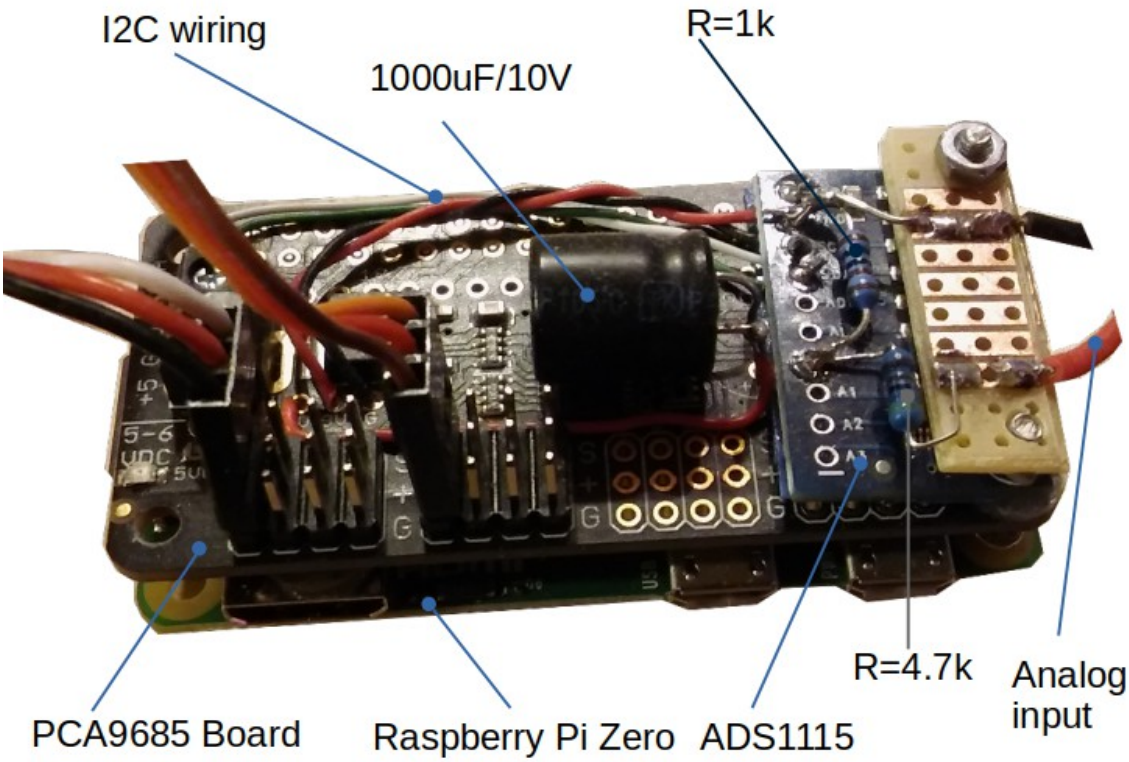

Figure 10: Receiver with analogue input

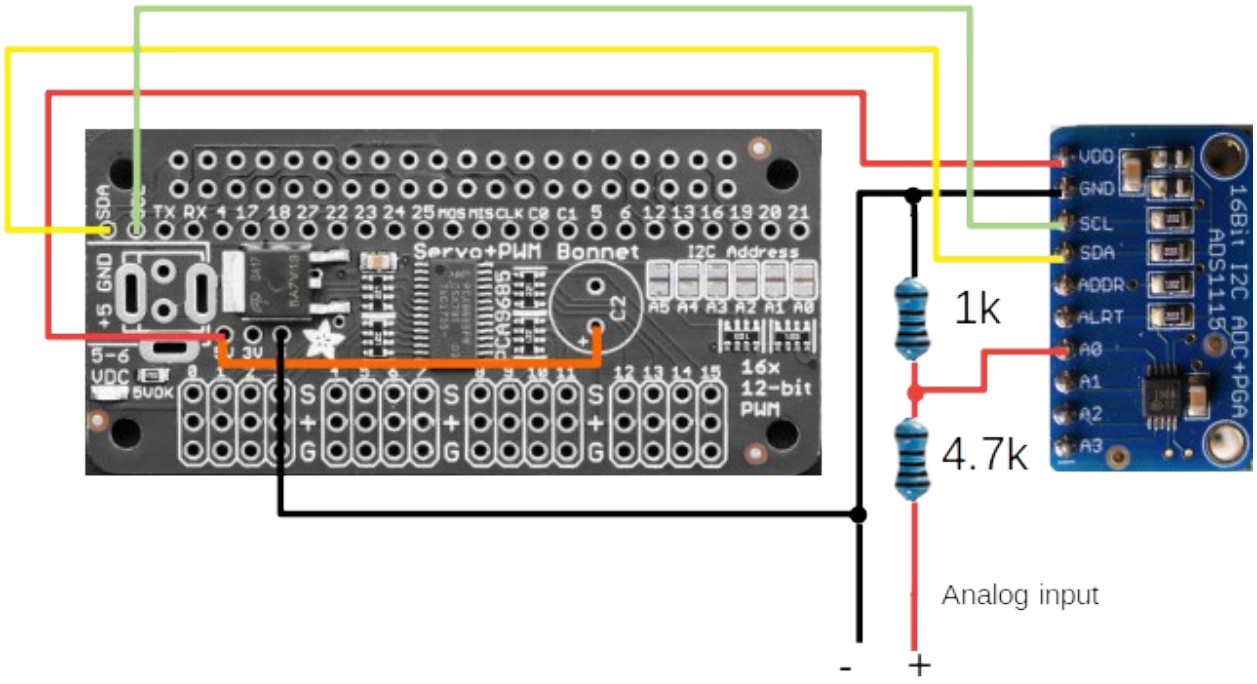

Bild 11: Wiring PCA9685 Board - ADS1115 Board