# Steuerung eines Bootes mit Raspberry Pi über WiFi

von Erich Boeck

## 0 Inhalt

# 1 Ein Boot mit dem Raspberry Pi fernsteuern

Alternativ zu einem passenden vorhandenen Boot kann das Sportboot MS Christine [1] verwendet werden (ca. 60 cm Länge). Es muss das Boot so verändert werden, dass der Raspberry Pi, die Akkus, eine kleine Interfaceschaltung, der Fahrtenregler, der Servo und die Kamera für den Raspberry Pi untergebracht werden können. Die WiFi-Antenne sollte über Wasser und nicht in einem Metallgehäuse sein.
Dazu wurde der Bauplan des Bausatzes für das Sportboot MS Christine (Original siehe [1]) entsprechend verändert (Schablonen siehe [2], bei Bedarf kann es von dort mit 100% und randlos ausgedruckt werden). Die elektronischen Teile sollten möglichst wasserdicht untergebracht werden.

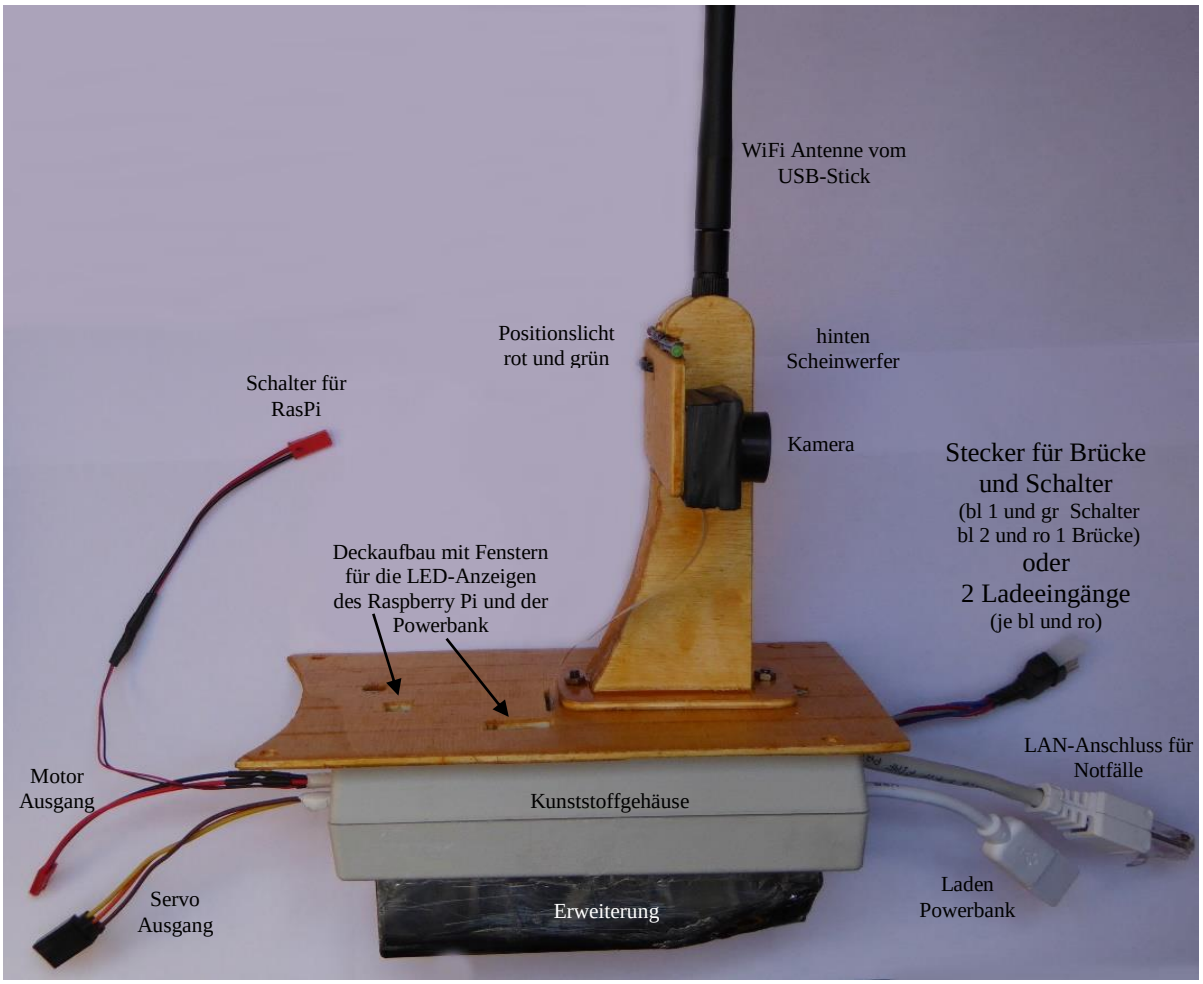



**Abb. 1: Pi 3B im Gehäuse mit Powerbank, Interfaceschaltung, Akkus und Fahrtenregler sowie darüber Aufbau für WiFi-Stick, Kamera und Scheinwerfer**

Die Steuerung ist in einem wasserdichten Gehäuse (Kunststoffgehäuse, 142 x 82,5 x 38 mm) mit Erweiterung des unteren Teils passend für den Bootskörper (aus Weißblech gelötet oder geklebt, Schablone siehe ebenfalls [2], mit 100% randlos ausdrucken).
Für die Erweiterung wurde ein entsprechender Ausschnitt im Boden des Gehäuses angebracht und die Erweiterung wasserdicht angeklebt.
Kabeldurchführungen können mit Abdichtmasse für Abwasserleitungen günstig gedichtet werden.
Wichtig ist, dass ein Laptop zur Verfügung steht, der auch im Freien (evtl. bei Sonnenschein) einen gut sichtbaren Bildschirm hat. Ein Touchscreen vereinfacht in diesem Fall die Bedienung.

# 2 Schaltung für die Steuerung

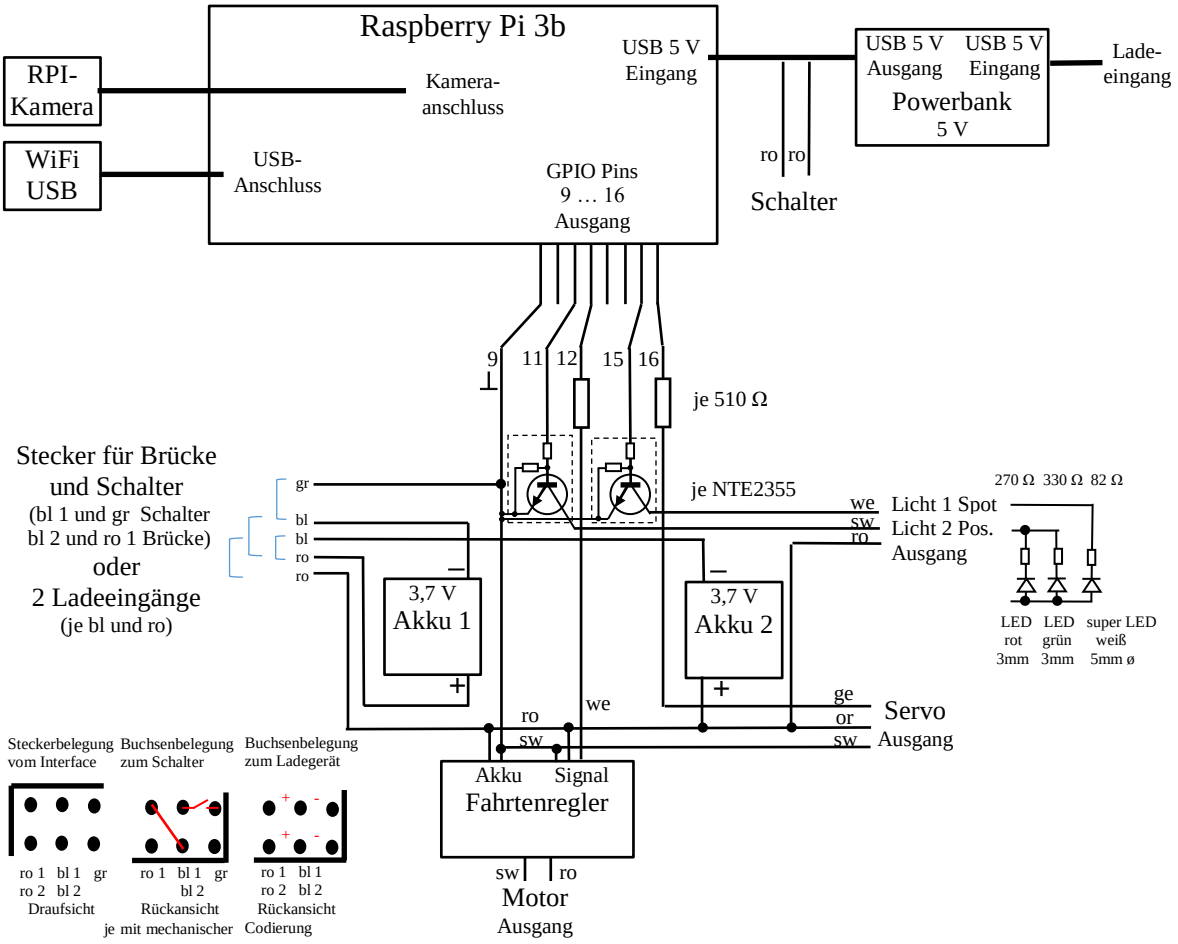



**Abb. 2: Schaltung der gesamten Steuerung**

Außer dem Einbau in das Gehäuse und die Verkabelung ist nur die Interfaceschaltung mit den 2 Schutzwiderständen, 2 Schalttransistoren und den Kabelanschlüssen zu bauen (siehe Abb. 2). An die Ausgänge für Licht kommen LED mit entsprechendem Vorwiderstand.

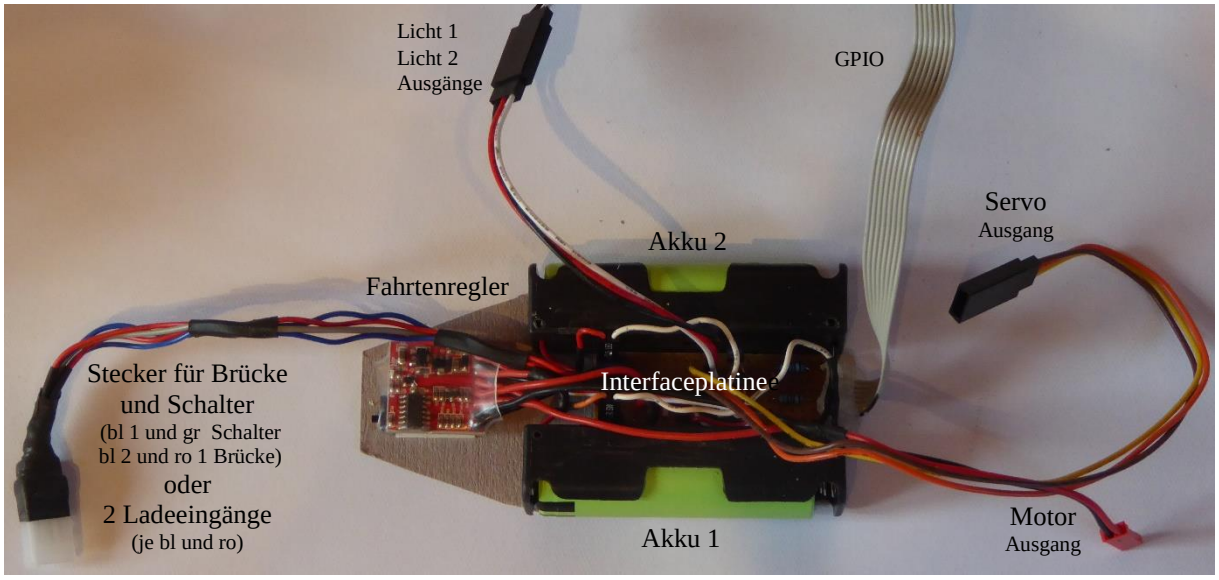



**Abb. 3: Interfaceschaltung mit angeschlossenem Fahrtenregler und den Ausgängen**

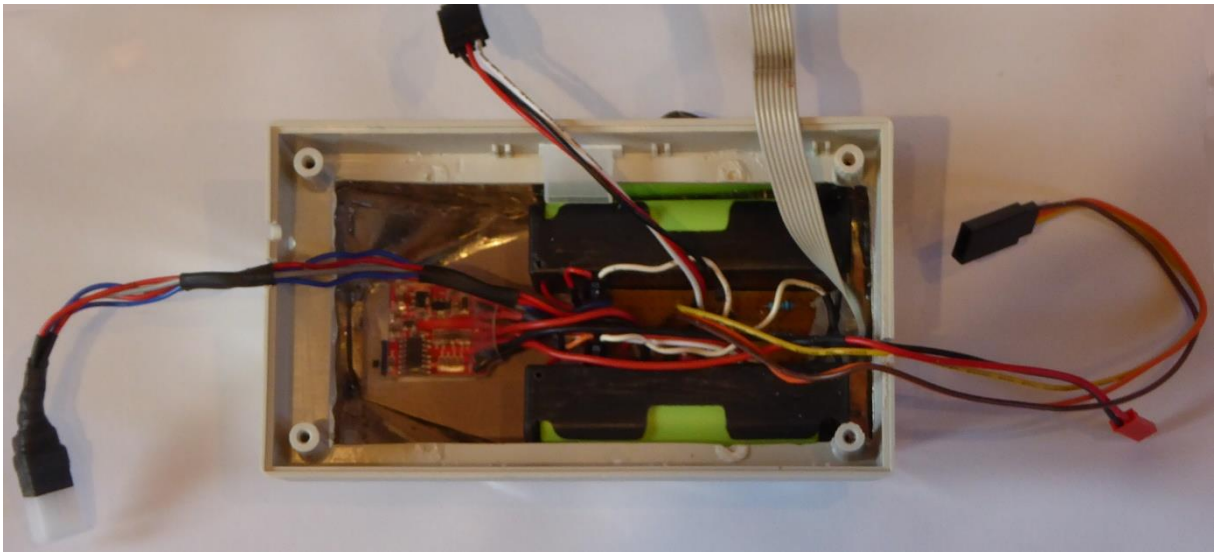
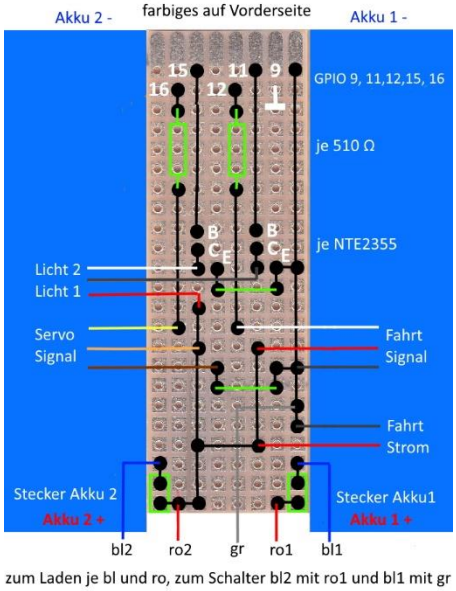
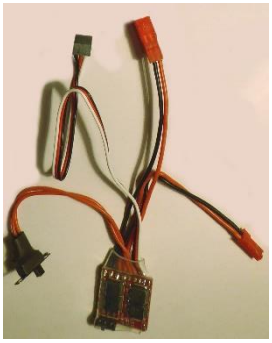


**Abb. 4: Akkus, Interfaceschaltung und Fahrtenregler im unteren Teil des Gehäuses**




Leiterplatte der Interfaceschaltung



Fahrtenregler für Motor:
Bürstenmotor ESC 20A Fahrregler Drehzahlregler für RC Auto Boot Hier mit Velleman DC-Motor, 6 V, 250 mA 14500 RPM (2,5–6V DC)

Servo für Ruder:
MG995 Micro Digital Servo Motor für RC Roboter Hubschrauber Flugzeug

Ladegerät mit angebauter Buchse für zwei separat zu ladende Akkus

Über den Akkus, der Interface-schaltung und dem Fahrtenregler ist die Powerbank untergebracht. Oben im Gehäuse ist der Raspberry Pi angeordnet.

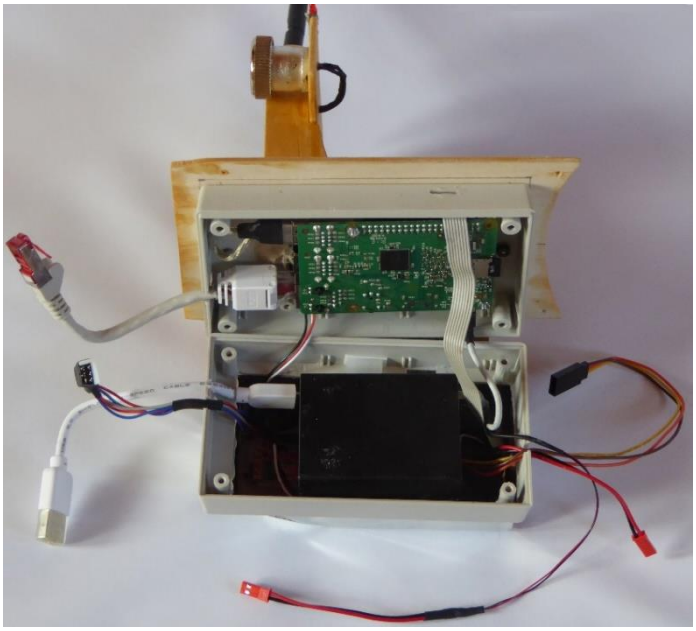


**Abb. 5: Details des Einbaus der Steuerung in das Gehäuse**

# 3 Konfiguration des Raspberry Pi

Einrichten des Raspberry Pi z.B. mit Raspbian Strech siehe [3].

## 3.1 GPIO des Raspberry Pi für Pulsweitenmodulation einsetzen [4]

Zum Standard auf dem Raspberry Pi haben sich für Lazarus [5] die C-Bibliotheken von [6] entwickelt. Lazarus und WiringPi ist im Raspbian-Repository enthalten. Die Aktualisierung bei neuen HW-Releases dürfte damit sichergestellt sein.

Ein bereits installiertes Paket wird geprüft mit:

```
gpio –v
```

Die verfügbare Version im Repository prüfen:

```
apt-cache show wiringpi
```

Installation aus dem Repository:

```
sudo apt-get install wiringpi
```

Testen:

```
gpio -v
gpio readall
```

**wiringPi.pas**

Der Pfad zur Unit muss für den Compiler auffindbar sein. Deshalb in Lazarus einstellen unter *„Projekt→ Projekteinstellungen→ Compilereinstellungen→ Pfade→ Andere Units→ "und hinzufügen „ .../myLib/wiringPi"*.

Einige Funktionen arbeiten mit Threads.

**wiringPi Setup**

Die Benutzung der Funktionen setzt zwingend den Aufruf einer von drei Setup-Funktionen voraus. Darin wird hauptsächlich das Zählschema der Pins für alle weiteren Funktionen festgelegt. WiringPi führt zur Kompatibilität der RPi-Revisionen ein eigenes Zählschema ein. Neben wiringPi lassen sich die Zählweise des Broadcom-Chips und die Pinzuordnung des Pi-Connectors initialisieren.

- function wiringPiSetup: longint; cdecl; external; //wiringPi-Schema
- function wiringPiSetupGpio: longint; cdecl; external; //Broadcom-Schema
- function wiringPiSetupPhys: longint; cdecl; external; //Pi-Connector

Der Setup sollte zum Programmstart bspw. in Formcreate des Hauptfensters aufgerufen werden.

| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 3.3v | | | 1 \|\| 2 | | | 5v | | |
| 2 | 8 | SDA.1 | IN | 1 | 3 \|\| 4 | | | 5v | | |
| 3 | 9 | SCL.1 | IN | 1 | 5 \|\| 6 | | | 0v | | |
| 4 | 7 | GPIO. 7 | IN | 1 | 7 \|\| 8 | 1 | ALT0 | TxD | 15 | 14 |
| | | 0v | | | 9 \|\| 10 | 1 | ALT0 | RxD | 16 | 15 |
| 17 | 0 | GPIO. 0 | IN | 0 | 11 \|\| 12 | 0 | IN | GPIO. 1 | 1 | 18 |
| 27 | 2 | GPIO. 2 | IN | 0 | 13 \|\| 14 | | | 0v | | |
| 22 | 3 | GPIO. 3 | IN | 0 | 15 \|\| 16 | 0 | IN | GPIO. 4 | 4 | 23 |
| | | 3.3v | | | 17 \|\| 18 | 0 | IN | GPIO. 5 | 5 | 24 |
| 10 | 12 | MOSI | IN | 0 | 19 \|\| 20 | | | 0v | | |
| 9 | 13 | MISO | IN | 0 | 21 \|\| 22 | 0 | IN | GPIO. 6 | 6 | 25 |
| 11 | 14 | SCLK | IN | 0 | 23 \|\| 24 | 1 | IN | CE0 | 10 | 8 |
| | | 0v | | | 25 \|\| 26 | 1 | IN | CE1 | 11 | 7 |
| 0 | 30 | SDA.0 | IN | 1 | 27 \|\| 28 | 1 | IN | SCL.0 | 31 | 1 |
| 5 | 21 | GPIO.21 | IN | 1 | 29 \|\| 30 | | | 0v | | |
| 6 | 22 | GPIO.22 | IN | 1 | 31 \|\| 32 | 0 | ALT0 | GPIO.26 | 26 | 12 |
| 13 | 23 | GPIO.23 | IN | 0 | 33 \|\| 34 | | | 0v | | |
| 19 | 24 | GPIO.24 | IN | 0 | 35 \|\| 36 | 0 | IN | GPIO.27 | 27 | 16 |
| 26 | 25 | GPIO.25 | IN | 0 | 37 \|\| 38 | 0 | IN | GPIO.28 | 28 | 20 |
| | | 0v | | | 39 \|\| 40 | 0 | IN | GPIO.29 | 29 | 21 |
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |

Bei uns werden genutzt:

WPi 0 ≙ Pin 11 ≙ BCM Pin 17 – Licht Position

WPi 1 ≙ Pin 12 ≙ BCM Pin 18 HW-PWM0

WPi 3 ≙ Pin 15 ≙ BCM Pin 22 – Licht Spot

WPi 4 ≙ Pin 16 ≙ BCM Pin 23 - SW-PWM

**Abb. 6: Screenshot nach Befehl „gpio readall"**

**Pin Modes**

Das Setzen und Lesen einzelner GPIO dürfte die einfachste Aufgabe sein. Vor einer IO-procedure muss der Pin-Mode definiert werden.

```
pinMode(Pin, mode);        //Pin entsprechend Setup
```

Folgende Modi sind möglich:

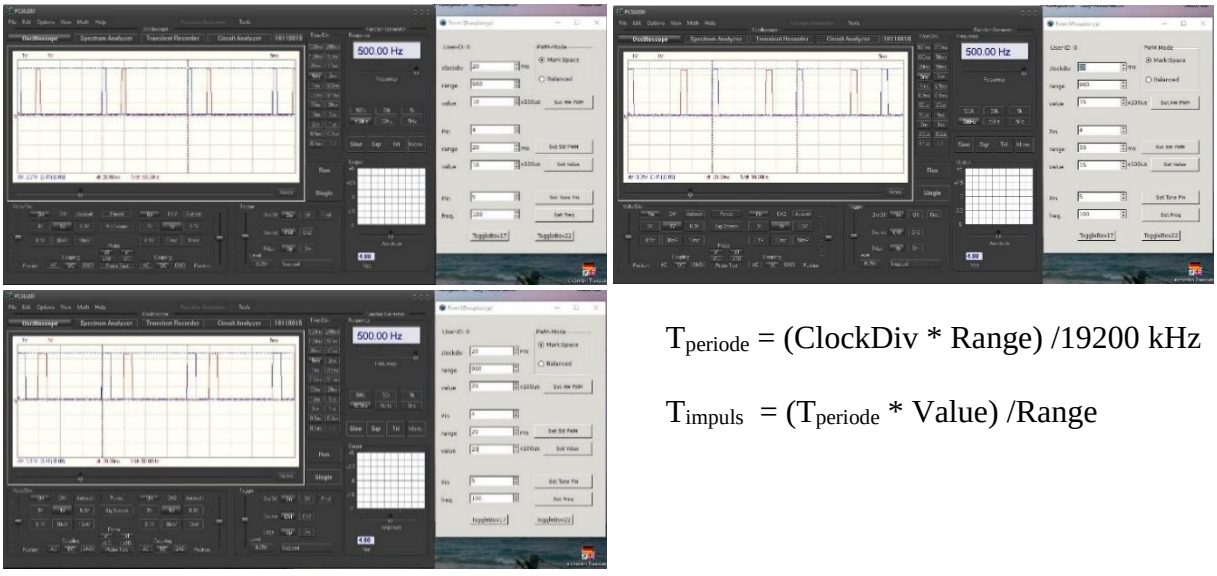
- 0: pmINPUT
- 1: pmOUTPUT
- 2: pmPWM_OUTPUT  //nur als root zulässig
- 3: pmGPIO_CLOCK  //nur als root zulässig
- 4: pmSOFT_PWM_OUTPUT
- 5: pmSOFT_TONE_OUTPUT
- 6: pmPWM_TONE_OUTPUT

### 3.1.1 Hardware - PWM

Hardware-PWM funktioniert derzeit nur mit **Root-Rechten**. Der Versuch den PinMode auf „*pmPWM_OUTPUT*“ zu setzen, führt ohne Root-Rechte zum Absturz.
Beim Rapsberry steht nur einer der beiden PWM-Kanäle zur Verfügung. PWM0 (BCM18) ist fest mit Pin 12 entspricht GPIO.1, WPi 1 verbunden (d.h. Pin ist 1).
//Init HW-PWM

```
pinMode(1, pmPWM_OUTPUT); //Pin 1, Mode switcht BCM.18 to ALT5, sudo needed
pwmSetMode(pwmMS);        //pwmMS -> Mark:Space || pwmBAL -> Balanced
pwmSetClock(40);          //clock=19.2MHz/ClockDiv (32bit >=2, hier ClockDiv=40)
pwmSetRange(9600);        //Range  (32bit hier 9600)
pwmWrite(1, 720);         //Pin 1, Value (32bit)(hier 15*48 entspricht 1,5ms)
```

**Timing der PWM-Modes mit o.g. Werten:**
Mark:Space  pwmMS bei f=50 Hz und 10, 15 sowie 20 je *100µs (blau); zusätzlich Software PWM (Rot):




$T_{periode}$ = (ClockDiv * Range) /19200 kHz

$T_{impuls}$ = ($T_{periode}$ * Value) /Range

**Abb. 7: Oszillogramme der Impulse vom Raspberry Pi Pin 12 (Hardware) und 16 (Software)**

### 3.1.2 Software - PWM

Zusätzlich ist in wiringPi Software - PWM deklariert. Es können alle freien GPIO angesteuert werden. Für jeden Pin wird ein separater Thread gestartet. Die Threads werden erst mit dem Hauptprogramm beendet. Ein vorzeitiges Ende gibt es nicht.

```
softPwmCreate(Pin, startValue, Range);
```

Die Funktion kann nur 1x je Pin aufgerufen und *Range* später nicht geändert werden.

$T_{impuls}$ wird zur Laufzeit geändert mit:

```
softPwmWrite(Pin, Value);
```

$T_{periode}$ = Range * 100 µs  bei uns 200 * 100 µs = 20 ms  d.h. 50 Hz
$T_{impuls}$ = Value * 100 µs  bei uns 10 bis 20 * 100 µs = 1 bis 2 ms
Die Genauigkeit ist nicht sehr hoch. Range-Werte ≥100 ergeben brauchbare Ergebnisse.

**Parameter für uns entsprechend der Signale für RC - Fahrtenregler und - Servo:**

| | | Timp/ms |
|---|---|---|
| Volle Fahrt voraus | VV | 1,0 |
| Halbe Fahrt voraus | HV | 1,2 |
| Langsame Fahrt voraus | LV | 1,3 |
| Kleine Fahrt voraus | KV | 1,4 |
| Halt | Halt | 1,5 |
| Kleine Fahrt zurück | KZ | 1,6 |
| Langsame Fahrt zurück | LZ | 1,7 |
| Halbe Fahrt zurück | HZ | 1,8 |
| Volle Fahrt zurück | VZ | 2,0 |
| Hart Backbord (ca. 60°) | BBh | 1,0 |
| Backbord 40° | BB40 | 1,2 |
| Backbord 25° | BB25 | 1,3 |
| Backbord 10° | BB10 | 1,4 |
| Mitschiffs (ca. 0°) | Mit | 1,5 |
| Steuerboard 10° | StB10 | 1,6 |
| Steuerboard 25° | StB25 | 1,7 |
| Steuerboard 40° | StB40 | 1,8 |
| Hart Steuerboard (ca. 60°) | StBh | 2,0 |

**Berechnung bei WiringPi**

```
HW-PWM      pinMode(1, pmPWM_OUTPUT)
//Pin = 1 ist WPi 1 ≙ Pin 12 ≙ BCM Pin 18
            pwmSetClock(ClockDiv)      //Wahl ClockDiv = 40  bei Clock = 19.2MHz
            pwmSetRange(Range)         //Wahl Range = 9600
                                       //mit Tper = (ClockDiv * Range) /19200kHz
                                       //d.h. Tper = (40 * 9600) /19200kHz = 20ms
            pwmWrite(1, Value)         //auch zum Ändern zur Laufzeit
                                       //mit Timp/ms = [(Tper/ms) * Value] /Range
            //d.h. Value = (Timp * 9600) /20ms = (Timp/ms) * 480
           //oder                       Value = (Timp/100µs)*48

SW-PWM      softPwmCreate(4, Value, Range)
                                       //Pin = 4 ist WPi 4 ≙ Pin 16 ≙ BCM Pin 23
                                       //Range erst nach reboot zu ändern!
                                       //mit Tper = 100 µs * Range
                                       //d.h. Range = 200
                                       //mit Timp = 100 µs * Value
                                       //d.h. Value = Timp /100 µs
            softPwmWrite(4, Value)     //auch zum Ändern zur Laufzeit
```

*Damit ergeben sich das Steuerprogramm und die Interfaceschaltung.*

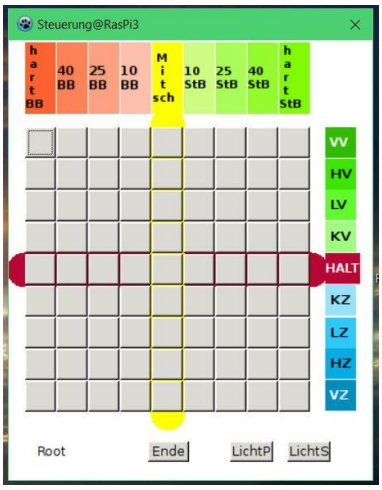
### 3.1.3 Programm zur Steuerung der Fahrt und Richtung mit Lazarus/free Pascal

Das Programm kann bei Lazarus nicht insgesamt einfach kopiert werden. Die Butten und andere Komponenten müssen aus dem Komponentenmenü auf dem Form - Formular platziert und im Objektinspektor konfiguriert werden, wonach schon vieles im Programm vorgetragen ist. Danach können die Ereignisbehandlungen mit Pascalbefehlen programmiert werden (Siehe Hinweise in der „unit Steuerfeld“).
(Bei Interesse am kompletten Programmpaket bitte eMail.)

```
unit Steuerfeld;

{$mode objfpc}{$H+}

interface

uses
  Interfaces, Classes, SysUtils, FileUtil, LResources, Forms, Controls,
  Graphics, Dialogs, Buttons, Menus, StdCtrls, baseunix, wiringpi;          //vervollständigen

type

  { TSteuerung }

  TSteuerung = class(TForm)       //Voreingestelltes evtl. entsprechend umbenennen

//Buttons
    Button1: TButton;        //Start
    Button2: TButton;        //Ende
    Button3: TButton;        //Licht Position          } Aus Komponentenmenü auf Form platzieren.
    Button4: TButton;        //Licht Spot
    Label1: TLabel;          //für Kontrolle auf root
    btnGenerate: TButton;  //zur automatischen Erzeugung des Arrays mit 9 x 9 Button (die Zeile kopieren)

//Label für Ruderstellung
    LBBh: TLabel;
    LBB40: TLabel;
    LBB25: TLabel;
    LBB10: TLabel;          //Aus Komponentenmenü auf Form platzieren,
    LMitS: TLabel;          //im Objektinspektor Left, Top, Height, Width und Color entsprechend Buttonarray und FarbeR konfigurieren
    LStB10: TLabel;         //und Caption sowie Font nach eigenem Wunsch einstellen.
    LStB25: TLabel;
    LStB40: TLabel;
    LStBh: TLabel;

//Label für Geschwindigkeit
    LVV: TLabel;
    LHV: TLabel;
    LLV: TLabel;
    LKV: TLabel;            //Aus Komponentenmenü auf Form platzieren,
    LHalt: TLabel;          //im Objektinspektor Left, Top, Height, Width und Color entsprechend Buttonarray und FarbeF konfigurieren
    LKZ: TLabel;            //und Caption sowie Font nach eigenem Wunsch einstellen.
    LLZ: TLabel;
    LHZ: TLabel;
    LVZ: TLabel;

//Bedienung (Prozeduren deklarieren)
    procedure btnClickEvent(Sender: TObject);   //Button für Ereignisse des Arrays (die Zeile kopieren)
    procedure Fahrt(kF: Integer; kR: Integer);     //Ruder und Geschwindigkeit einstellen und anzeigen (die Zeile kopieren)
    procedure FormCreate(Sender: TObject);      //für Voreinstellungen
    procedure Button1Click(Sender: TObject);    //für Bedienung aktivieren
    procedure FormPaint(Sender: TObject);       //für alles anzeigen
    procedure Button3Click(Sender: TObject);    //für Licht Position          } Je im Objektinspektor unter Ereignisse auswählen.
    procedure Button4Click(Sender: TObject);    //für Licht Scheinwerfer (Spot)
    procedure Button2Click(Sender: TObject);    //für alles beenden
    procedure FormDestroy(Sender: TObject);     //für alles beenden mit "X"

  private

  public

  end;
```

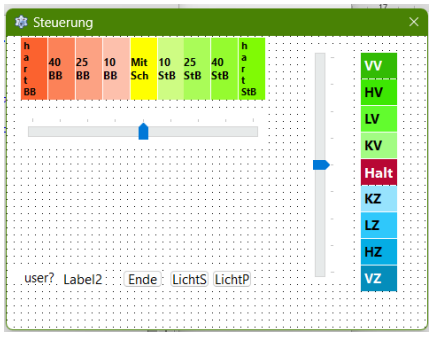
```
var
  Steuerung: TSteuerung;                  //Form (Fenster) deklarieren (wird voreingestellt).
  FarbeF: Array[0..8] of TColor;
  FarbeR: Array[0..8] of TColor;
  PosF:   Array[0..8] of Integer;
  PosR:   Array[0..8] of Integer;
  TimpF:  Array[0..8] of Integer;         //notwendige globale Variablen (die Zeilen kopieren)
  i: Integer;
  PosF1: Integer=100;
  PosR1: Integer=31;

implementation

{$R *.lfm}

{ TSteuerung }

procedure TSteuerung.btnClickEvent(Sender: TObject);  //Ereignisse der Button des Arrays verarbeiten
var z: Integer;
   kF: Integer;
   kR: Integer;
begin
 z:= (Sender as TControl).tag;
 kF:= z div 10;                           //Division mit Ergebnis als abgerundete ganze Zahl
 kR:= z-kF*10;
 //Label1.Caption:='kR=' + IntToStr(kR) + '  kF=' + IntToStr(kF);  //Zeile für Kontrolle unkommentieren
 Fahrt(kF,kR);
end;

procedure TSteuerung.Fahrt(kF: Integer; kR: Integer);  //Ruder und Geschwindigleit einstellen
var TimpHW: Integer;
begin
  TimpHW:= TimpF[kF]*48;          //TimpF in 100 us (siehe auch Array)
                                  //Timp=Tper*TimpHW/Range d.h. 1...2 ms
  pwmWrite(1, TimpHW);            //(Pin,TimpHW) mit TimpHW=[TimpF(kF)/100 us]*48
  with Canvas do begin                                                    beide Prozeduren kopieren
    Pen.Color := clWhite;         //Löschen des Vorherigen
    FillRect(0, 70, 309, 382);
    Pen.Width := 34;
    Pen.Color := FarbeF[kF];      //kF wählen
    MoveTo(15,PosF[kF]);          //markieren
    LineTo(295,PosF[kF]);
  end;
  //Verändern TimpSW der SW-PWM
  //TimpSW in 100 us d.h. 1...2 ms (siehe Array TimpF)
  softPwmWrite(4, TimpF[kR]);     //(Pin,TimpSW) mit TimpSW = TimpF[kR]
  with Canvas do begin
    Pen.Width := 34;
    Pen.Color := FarbeR[kR];      //kR wählen
    MoveTo(PosR[kR],83);          //markieren
    LineTo(PosR[kR],364);
  end;
end;

//Ab hier für alle Ereignisse nach Erzeugung im Objektinspektor (siehe oben Typedeklaration) die Inhalte kopieren.

procedure TSteuerung.FormCreate(Sender: TObject);
var uid: longint;
begin                                     //Wenn Form „Steuerung“ erzeugt wird ausführen.
   uid:= fpGetUid();                      //Test ob Rootrechte
   if uid  = 0 then label1.Caption := '   Root'
   else label1.Caption := 'kein Root';
   if uid <> 0 then
    begin
      Button1.Enabled := false;
      Button2.Enabled := true;
    end;
   //Setup in WiringPi-Numbering
   if uid  = 0 then
    begin
      //Setup in WiringPi-Numbering
      wiringPiSetup();
      //Initialisierung HW-PWM WPi 1  Pin 12 und Voreinstellung auf "Halt" entspricht 1.5 ms Impuls bei 20 ms Periode
      pinMode(1, pmPWM_OUTPUT);   //schaltet nach ALT5
      pwmSetMode(pwmMS);                  //PWMMode = pwmMS
      pwmSetClock(40);                    //Clock = 19.2MHz /40   (40 = ClockDiv)
     //ClockDiv und Range so zu wählen, dass Timp genau einstellbar wird.
```

```
    pwmSetRange(9600);                  //Tper=40*9600/19200 kHz=20 ms
                                        //Timp=Tper*TimpHW/Range d.h. 1,5 ms, d,h, Voreinstellung Halt
    pwmWrite(1, 720);                   //(Pin,TimpHW) mit TimpHW=TimpF(kF)(in 100 us)*48
    //Initialisierung SW-PWM WPi 4  Pin 16
    //TimpSW = 15 in 100 us (d.h. 1,5 ms) = Voreinstellung Mittschiffs und Tperiode:= 200 in 100 us (d.h. 20 ms)
    softPwmCreate(4, 15, 200);          // (Pin,TimpSW,Tperiode)
    //Initialisieren Licht
    pinMode(0,pmOUTPUT);
    pinMode(3,pmOUTPUT);
    //Licht aus
    digitalWrite(0,levLOW);
    digitalWrite(3,levLOW);
   end;
end;

//Bedienung
procedure TSteuerung.Button1Click(Sender: TObject);  //Start, zum aktivieren der Steuerung
var
  btn:TButton;
  x: Integer;
  y: Integer;
  W: Integer = 30;
  H: Integer = 30;
  T: Integer = 85;
  L: Integer = 16;
  FS: Integer = 12;
begin
  Button1.Visible:=false;                //Button Start unsichtbar
  Button2.Visible:=true;                 //Button Ende sichtbar
  Button3.Visible:=true;                 //Button Licht Position sichtbar
  Button4.Visible:=true;                 //Button Licht Spot sichtbar
  For y:= 0 to 8 do                      //Buttons von links nach rechts erzeugen
   begin
   For x:= 0 to 8 do                     //Buttons von oben nach unten erzeugen
    begin
     btn:=TButton.create(nil);
     btn.parent:=Steuerung;
     btn.Font.Size:=FS;
     btn.Caption:=' ';
     btn.Name:='Button' + IntToStr(x) + IntToStr(y);
     btn.Tag:= 10*x+y;
     btn.Width:=W;
     btn.Height:=H;
     btn.Top:=T;
     btn.left:=L;
     btn.OnClick := @btnClickEvent;
     T:=T + 31;
    end;
   T:= 85;
   L:= L + 31;
  end;
end;
```




```
procedure TSteuerung.FormPaint(Sender: TObject);      //beim neu Fenster darstellen neu zeichnen
begin
  //Halt Mittschiffs voreinstellen
  Fahrt(4, 4);
  //Licht aus
  digitalWrite(0,levLOW);
  digitalWrite(3,levLOW);
end;

procedure TSteuerung.Button3Click(Sender: TObject);
var v1 : tLevel;
begin
  //Licht Position ein/aus
  v1:= digitalRead(0);
  if v1 = levHIGH then digitalWrite(0,levLOW) else digitalWrite(0,levHIGH);
end;

procedure TSteuerung.Button4Click(Sender: TObject);
var v1 : tLevel;
begin
  //Licht Spot ein/aus
  v1:= digitalRead(3);
  if v1 = levHIGH then digitalWrite(3,levLOW) else digitalWrite(3,levHIGH);
end;
```

```
procedure TSteuerung.Button2Click(Sender: TObject);   //bei Button Ende
begin
  //TimpHW:= aus
  pwmWrite(1, 0);
  //TimpSW:= aus
  softPwmWrite(4, 0);
  //Licht aus
  digitalWrite(0,levLOW);
  digitalWrite(3,levLOW);
  //Programm beenden
  Steuerung.Close;
end;

procedure TSteuerung.FormDestroy(Sender: TObject);    //bei Fenster mit "X" schließen
begin
  //Programm beenden
  //TimpHW:= aus
  pwmWrite(1, 0);
  //TimpSW:= aus
  softPwmWrite(4, 0);
  //Licht aus
  digitalWrite(0,levLOW);
  digitalWrite(3,levLOW);
end;


//Hauptprogramm: Arrays für Farben, Positionen und Impulszeiten (alles kopieren)

begin
 FarbeF[0]:= $0004BB32;
  FarbeF[1]:= $0003E73D;
  FarbeF[2]:= $002EFC62;
  FarbeF[3]:= $0083FCA2;
  FarbeF[4]:= $003407B8;
  FarbeF[5]:= $00FDE397;
  FarbeF[6]:= $00FBC82F;
  FarbeF[7]:= $00E4AD05;
  FarbeF[8]:= $00BB8D04;
  FarbeR[0]:= $002F62FB;
  FarbeR[1]:= $005882FC;
  FarbeR[2]:= $0083A2FC;
  FarbeR[3]:= $00ACC0FD;
  FarbeR[4]:= clYellow;
  FarbeR[5]:= $0083FCCE;
  FarbeR[6]:= $0058FCAA;
  FarbeR[7]:= $002FFB95;
  FarbeR[8]:= $0005FA80;
  For i:= 0 to 8 do
    begin
      PosF[i]:= PosF1;
      PosF1:=PosF1+31;
      PosR[i]:= PosR1;
      PosR1:=PosR1+31;
    end;
  TimpF[0]:= 10;    //wird 1,0 ms
  TimpF[1]:= 12;    //wird 1,2 ms
  TimpF[2]:= 13;    //wird 1,3 ms
  TimpF[3]:= 14;    //wird 1,4 ms
  TimpF[4]:= 15;    //wird 1,5 ms
  TimpF[5]:= 16;    //wird 1,6 ms
  TimpF[6]:= 17;    //wird 1,7 ms
  TimpF[7]:= 18;    //wird 1,8 ms
  TimpF[8]:= 20;    //wird 2,0 ms
 end.
```




Alternativ können auch gut zwei TrackBar zur Fahrt- und Rudersteuerung genutzt werden.

Das Programm muss mit Lazarus auf den Raspberry Pi erarbeitet werden, weil es Hardware-zugriffe auf den Pi beinhaltet. Bei Bearbeiten auf Windows müssen alle Befehle mit Hardwarezugriffen auskommentiert werden (dann verschiedene Versionen von Lazarus beachten).

## 3.2 Kameralivestream [7]

! nur rpi-Kameramodul !

**Kamera einrichten:**
sudo apt-get install v4l-utils
Damit die Kamera angezeigt wird:
sudo modprobe bcm2835-v4l2
Ist nur eine einzige Webcam angeschlossen, sollte Device /dev/video0 angegeben sein.
Angeschlossene Videogeräte / Kameras anzeigen:
ls /dev/video*

Damit dies Modul bei einem Neustart ebenfalls wieder direkt geladen wird, kann man es auch noch direkt in die Modulliste eintragen:
sudo nano /etc/modules
Am Ende in die Datei eintragen
sudo bcm2835-v4l2
Speichern mit STRG + O, Beenden mit STRG + X.

Kamera ansehen:
sudo vcgencmd get_camera
Kamera Details ansehen:
sudo v4l2-ctl -V

**Direkt mit raspivid streamen:** [8]
direct tcp stream:
Raspi mit rpi-Kameramodul
Im Terminal eingeben: sudo raspivid -fps 12 -w 640 -h 480 -t 0 -l -o tcp://0.0.0.0:5000
Windows 10 mit SMPlayer - dazu ist der SMPlayer zusätzlich zu installieren [9]:
(Nur der SMPlayer unterstützt direkt den Codec von raspivid. Somit kein Umcodieren und also geringe Prozessorlast.)
Im SMPlayert aufrufen: tcp://192.168.1.20:5000 (IP des Raspi einsetzen!)
**Menüeintrag** für Raspi zum Starten:
Datei /home/pi/.local/share/applications/Kamera.desktop anlegen mit

```
[Desktop Entry]
Type=Application
Icon=gpicview
Name=Kamerastream
Name[de]=Kamerastream
GenericName=Kamerastream
GenericName[de]=Kamerastream
Comment=Kamerastream starten
Comment[de]=Kamerastream starten
Categories=
Exec=sudo raspivid -fps 12 -w 640 -h 480 -t 0 -l -o tcp://0.0.0.0:5000
StartupNotify=true
Terminal=false
```

Nach Start leuchtet Kameramodul und für Windows 10 mit SMPlayer
tcp://192.168.1.20:5000 (IP des Raspi einsetzen!)
aufrufen. Nach Beenden des SMPlayer wird raspivid beendet und es erlischt die Kamerakontrollleuchte.

Alternativ kann „rpi_camera_web_streaming.py“ genutzt werden. Das reagiert sogar deutlich schneller (siehe Anhang).

## 3.3 WiFi Accesspoint einrichten [10]

Raspberry Pi 3 hat ein WLAN zusätzlich ist ein USB WLAN-Stick mit Antenne sinnvoll (dazu evtl. spezieller Treiber notwendig).
Es sind DNSMasq and HostAPD zu installieren:
sudo apt install dnsmasq
sudo apt install hostapd

1. Statische IP ist für wlan1 für den Accesspoint erforderlich für eth0 und wlan0 nicht unbedingt dafür aber in /etc/ wpa_supplicant/ eine wpa_supplicant.conf (nur Original), wpa_supplicant.conf-wlan1(wie Original) und wpa_supplicant.conf-wlan0 (mit zusätzlich:

```
network={
    ssid="ssid des zu verbindenden Wlan"
    psk="Passwort des zu verbindenden Wlan"
}
```

anlegen. Konfiguration der statischen IP
sudo nano /etc/dhcpcd.conf
am Ende eintragen:

```
interface wlan1
static ip_address=192.168.4.1/24   # für wlan1 keine weiteren Eintragungen, wenn eth0, wlan0 auch static ip dafür auch Gateway und DNS
```

Speichern mit STRG + O, Beenden mit STRG + X und Restart dhcpcd daemon zum starten der neuen wlan1 Konfiguration:
sudo service dhcpcd restart
Dann sicherstellen, dass das Ethernet-Interface (eth0), wlan0 als auch der WLAN-Adapter (wlan1) funktionieren und vorhanden sind:
ip l

2. Konfiguration des DHCP Servers (dnsmasq)
Zuerst Umbenennen der originalen Konfigurationsdatei dann neue Konfigurationsdatei erzeugen:
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
sudo nano /etc/dnsmasq.conf
Nun Eintragen und Speichern mit STRG + O, Beenden mit STRG + X:

```
no-dhcp-interface=eth0
no-dhcp-interface=wlan0
interface=wlan1
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h  #wlan1 für IP Adressen von 192.168.4.2 bis 192.168.4.20, mit lease time 24 h.
dhcp-option=option:dns-server,192.168.4.1  #hier DNS für Accesspoint wlan1 nach static ip für wlan1
```

Vor der Inbetriebnahme empfiehlt es sich, die Konfiguration zu testen.

```
dnsmasq --test -C /etc/dnsmasq.conf
```

Die Syntaxprüfung sollte mit "OK" erfolgreich sein.

DNSMASQ neu starten:

```
sudo systemctl restart dnsmasq
```

DNSMASQ-Status anzeigen:

```
sudo systemctl status dnsmasq
```

DNSMASQ beim jedem Systemstart starten:

```
sudo systemctl enable dnsmasq
```

3. Konfiguration der Accesspoint Software (hostapd):

```
sudo nano /etc/hostapd/hostapd.conf
```

Information zur Konfigurationsdatei hinzufügen (Speichern und Schließen mit Strg + O, Return, Strg + X):

```
# Schnittstelle und Treiber
interface=wlan1
#driver=nl80211                (wird normalerweise automatisch erkannt deshalb als Kommentar)
# WLAN-Konfiguration
ssid= xxxxx                    (hier Name des Netzwerks)
channel=3                      (oder 1 bis 13)
hw_mode=g
ieee80211n=1
ieee80211d=1
country_code=DE
wmm_enabled=1
# WLAN-Verschlüsselung
auth_algs=1
wpa=2
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
wpa_passphrase=xxxxxxxx      (hier Passphrase/Password vergeben)
```

WLAN-AP-Host-Konfiguration prüfen und dazu in Betrieb nehmen (hostapd):

```
sudo hostapd -dd /etc/hostapd/hostapd.conf
```

Wenn „ok“ läuft die Konfiguration durch aber das Programm wird nicht beendet. Es sollten die Zeilen

```
...
wlan1: interface state COUNTRY_UPDATE->ENABLED
...
wlan1: AP-ENABLED
...
```

vorhanden sein.

Jetzt kann auch WLAN-AP getestet werden. Dazu mit einem WLAN-Client das WLAN finden und sich dort anmelden.
Mit "Strg + C" kann man die laufende hostapd-Instanz beenden.

Dem System den Speicherort der Konfigurationsdatei mitteilen:

```
sudo nano /etc/default/hostapd
```

Darin ergänzen wir folgende Parameter:

```
RUN_DAEMON=yes
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Anschließend speichern und schließen mit Strg + O, Return, Strg + X. Enable und start hostapd:

```
sudo systemctl unmask hostapd
sudo systemctl start hostapd
sudo systemctl enable hostapd
```

Status überprüfen:

```
sudo systemctl status hostapd
```

Weil die Datei „hostapd.conf“ das WLAN-Passwort im Klartext enthält, sollten nur die Benutzer „root“ (und „Pi“) Leserechte haben.

```
sudo chmod 600 /etc/hostapd/hostapd.conf
```

4. Hinzufügen routing and masquerade

Editieren  sudo nano /etc/sysctl.conf und
die folgende Zeile ergänzen

```
net.ipv4.ip_forward=1
```

danach schließen mit Strg + O, Return, Strg + X.

Hinzufügen von masquerade for outbound traffic des eth0 und wlan0:

```
sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -t nat -A  POSTROUTING -o wlan0 -j MASQUERADE
```

und Speichern der Festlegung mit:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Editieren

```
sudo nano /etc/rc.local
```

und hinzufügen vor "exit 0" zum Installieren der Festlegung beim Booten.

```
iptables-restore < /etc/iptables.ipv4.nat
```

danach schließen mit Strg + O, Return, Strg + X.

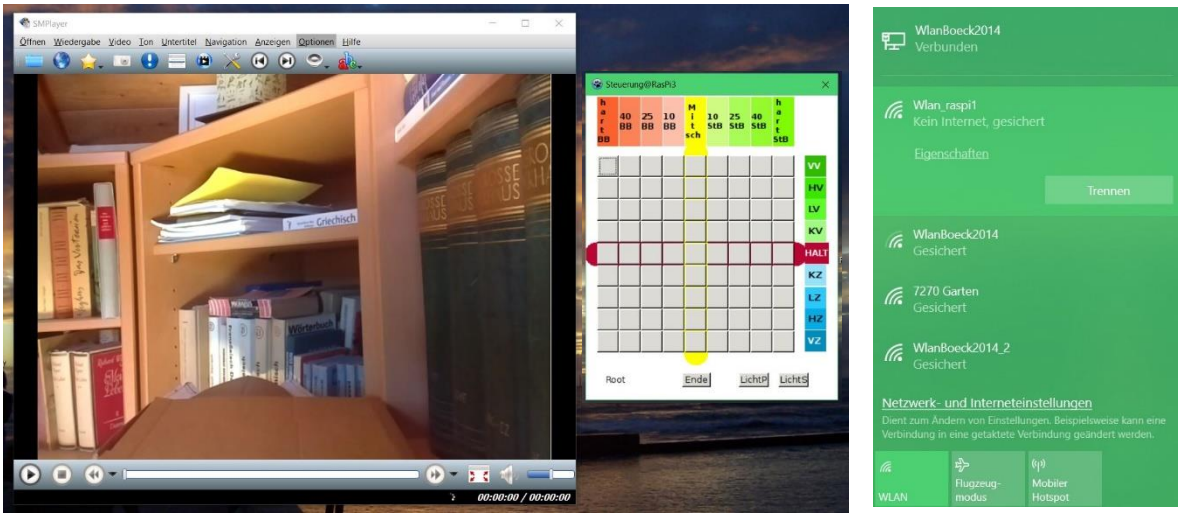



**Abb. 8: Screenshot vom Laptop nach Verbinden mit dem WiFi des Raspberry Pi**

Kamerastream im SMPlayer und Fenster (Form) der Steuerung nach einer Verbindung mit PuTTY über SSH im X11-Fenster von VcXsrv. Durch die Matrixform können zum besseren Manövrieren Geschwindigkeit (senkrecht) und Richtung (waagerecht) gleichzeitig gesteuert werden.

# 4 Aufbau

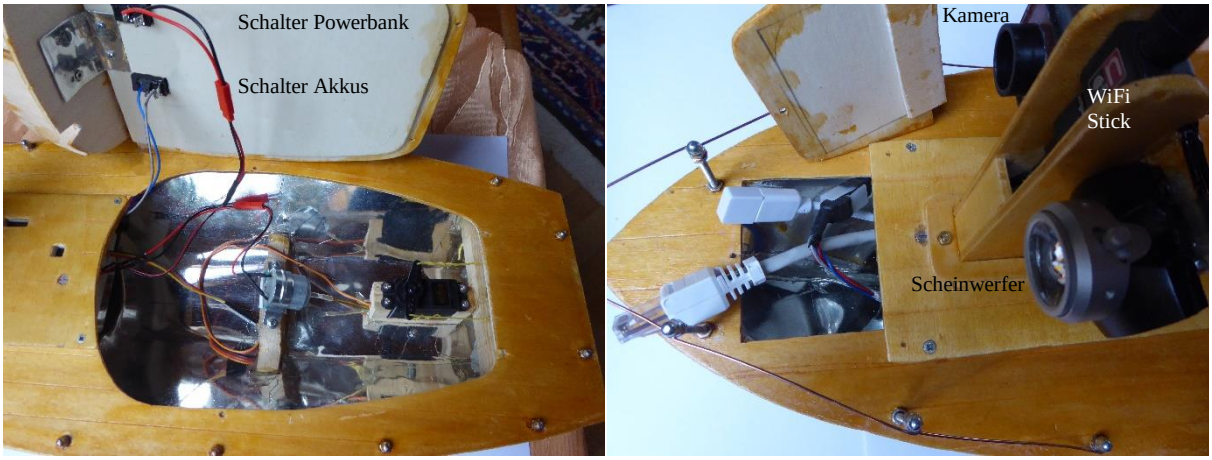
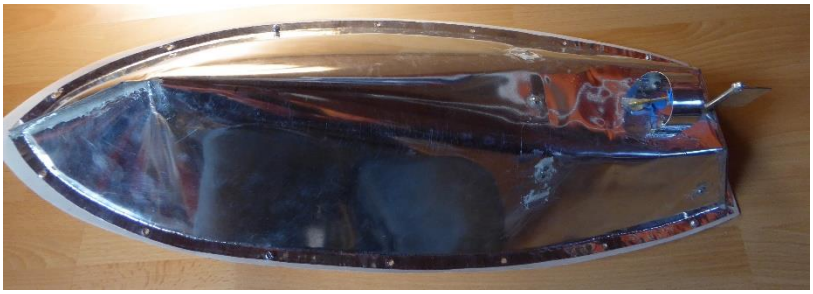



**Abb. 9: Hecköffnung mit Motor, Servo und den Schaltern, Bugklappe mit LAN-, Lade- und Schalteranschlüssen**

**Abb. 10: Gesamtansicht**



Heck mit dem Ruder und
Abdeckung des Propellers

**Abb. 11: Unterseite von MS Christine**

Beim Bau des Bootes MS Christine [1] ist vor allem nach der Originalbauanleitung vorzugehen. Es sind weiterhin die Veränderungen zur Anpassung (siehe z.B. Vergrößerungen am Heck [2]) sowie individuelle Anpassungen entsprechend der eingesetzten Gehäuse für die Elektronik zu berücksichtigen.

**Abb. 12: Erster Test im Wasser**
Es liegt stabil im Wasser und funktioniert bei einfachen Tests.

Der Farbanstrich auf dem Weißblech erfordert
1. ein Entfetten und feines Anschleifen des Weißbleches (sonst perlt der Lack zusammen),
2. eine Grundierung mit Rostschutz (da evtl. mal zu tiefes Schleifen sonst Rostspuren im Lack ergibt) und
3. ein feines Anschleifen jeder Lackschicht vor dem Überstreichen.

Bei ersten Testfahrten empfiehlt sich eine Schnur für alle Fälle.

# 5 Anhang

## rpi_camera_web_streaming.py [11]

```
import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>picamera MJPEG streaming demo</title>
</head>
<body>
<h1>PiCamera MJPEG Streaming Demo</h1>
<img src="stream.mjpg" width="640" height="480" />
</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
```

```
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8081)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()
```

# 6 Literaturverzeichnis


[1] „Sportboot 'MS Christine',“ Opitec, [Online]. Available: https://www.opitec.de/werkpackungen/holzbausaetze-/boote-schiffe--/sportboot-ms-christine.html?pgNr=2. [Zugriff am 08 04 2020].

[2] E. Boeck, „Schablonen Boot,“ Schablonen Boot.pdf, 21.04.2021.

[3] „Raspberry Pi: Erste Schritte bei der Installation,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/raspberry-pi/1905261.htm. [Zugriff am 14 04 2020].

[4] „Grundlagen der Schnittstellenprogrammierung,“ [Online]. Available: https://www.bw38.de/lazarusbasics. [Zugriff am 07 11 2019].

[5] „Lazarus on Raspberry Pi/de,“ [Online]. Available: https://wiki.lazarus.freepascal.org/Lazarus_on_Raspberry_Pi/de. [Zugriff am 15 03 2018].

[6] „wiringPi-Projekt GPIO Interface library for the Raspberry Pi,“ [Online]. Available: http://wiringpi.com/. [Zugriff am 08 04 2020].

[7] „Raspberry Pi/Camera streaming,“ [Online]. Available: http://www.netzmafia.de/skripten/hardware/RasPi/kamera/index.html. [Zugriff am 10 12 2019].

[8] „Raspberry Pi/Camera streaming,“ [Online]. Available: https://wiki.marcluerssen.de/index.php?title=Raspberry_Pi/Camera_streaming. [Zugriff am 08 04 2020].

[9] „SMPLAYER,“ [Online]. Available: https://www.smplayer.info/. [Zugriff am 08 04 2020].

[10] „Raspberry Pi als WLAN-Router einrichten,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/raspberry-pi/2002171.htm. [Zugriff am 21 01 2020].

[11] D. v. braspi, „Anschließen und Betreiben einer Raspberry Pi Kamera,“ [Online]. Available: https://www.braspi.de/2017/08/10/anschliessen-und-betreiben-einer-raspberry-pi-kamera/. [Zugriff am 25 06 2024].