Steuerung Raspberry Pi über WebApp mit Lazarus

von Erich Boeck

0 Inhalt

1 Erstellen einer WebApp mit Lazarus

Für eine WebAbb reicht es in ein funktionierendes Steuerprogramm (z.B. von [1]), eine unit mit „Datei>Neu>Web Module“ einzufügen. Dazu aus fpWeb die Componente FPHttpServer platzieren. Wenn alles richtig konfiguriert ist, können aus einer Webseite mit

<form name="form1" action="http://<WiFi IP Adresse des RasPi>:8080 " method="post" target="frame1" autocomplete="off">

die Daten gelesen und dem Program übergeben werden. Durch das iframe wird die Seite nicht neu geladen, was durch das Hintergrundbild (stream von RasPi-Kamera) sehr lange dauert kann und alles blockieren würde.
Der Nachteil ist, das man sich am Raspberry Pi (z.B. über SSH) anmelden muss, weil ein Start beim booten keine Form-Fenster ermöglicht.

Deshalb wird im Folgenden ohne Form Elemente programmiert.

Die notwendige Webseite mit CSS und JavaScript wird mit der Hand erstellt. Es werden Schieberegler (type="range") genutzt und das Hintergrundbild vom Stream der Webkamera.

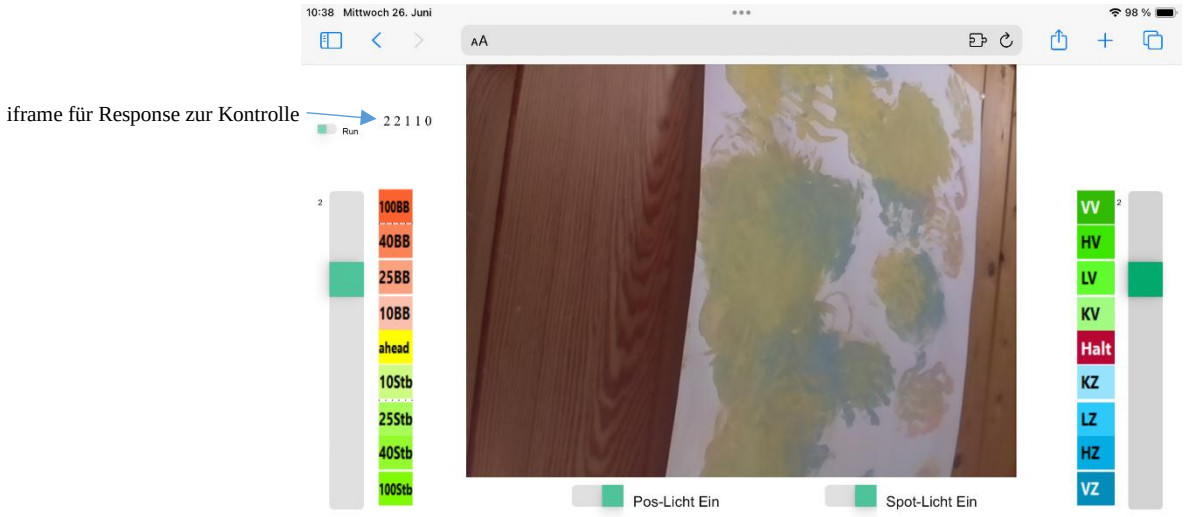


Abb. 1: Beispiel Webseite (Screenshot vom iPad mit Safari)

Der linke Schieberegler (von 0 bis 8) ist für das Ruder, der rechte Schieberegler für die Fahrt, die unteren Schieberegler (von 0 bis 1) sind Schalter für das Licht des Bootes. Der kleine Schalter oben setzt bei 1 alles zurück und startet einen Shudown.
Erprobt mit Pi OS bullseye, Lazarus 2.0.10 und fpc 3.2.0. Es funktioniert zumindest mit Firefox, Edge und Safari.

2 Schaltung für die 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 (Abb. 2). An die Ausgänge für Licht kommen LED mit entsprechendem Vorwiderstand.

Das Gehäuse sollte sicherheitshalber wasserdichtsein.

Der Aufbau entspricht [1]

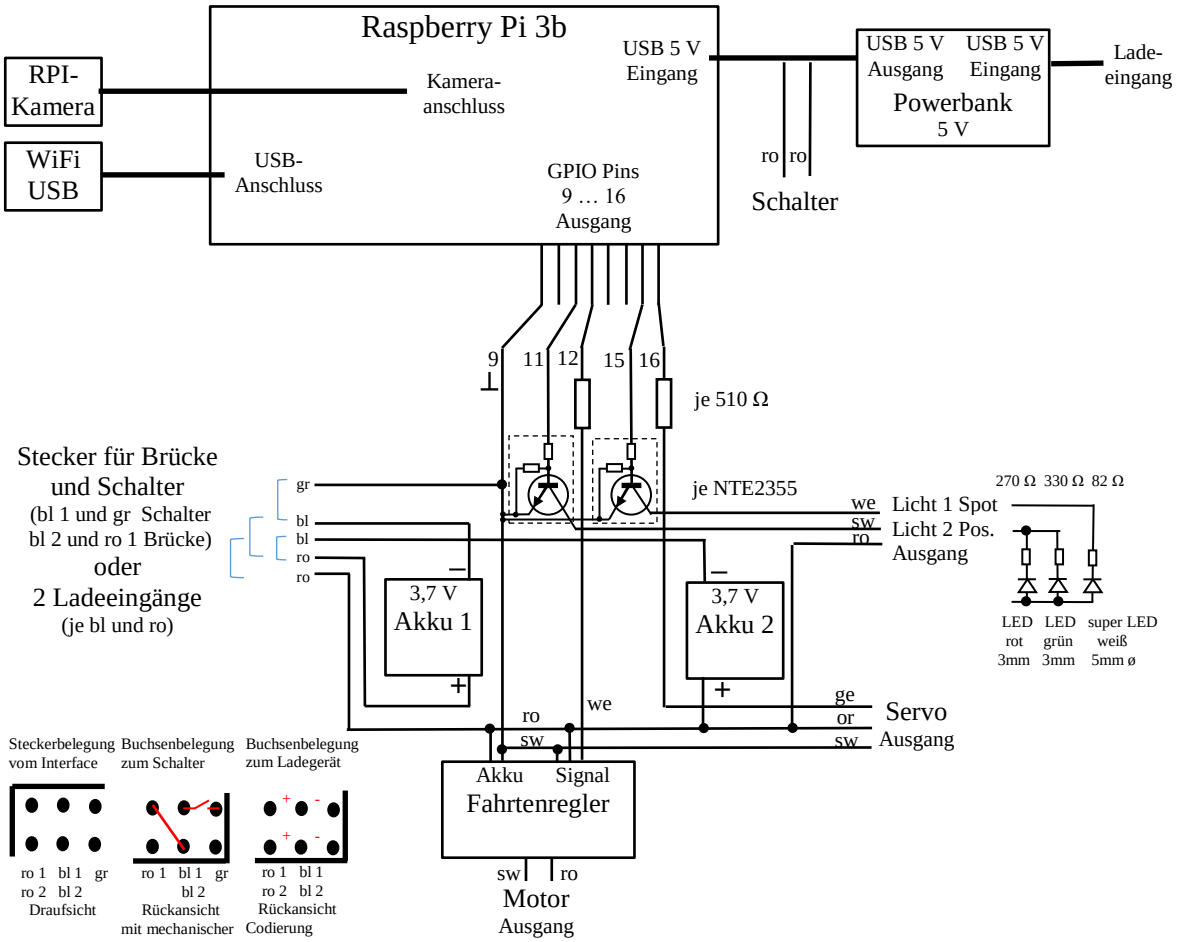


Abb. 2: Schaltung der gesamten Steuerung

3 Konfiguration des Raspberry Pi

Einrichten des Raspberry Pi z.B. mit Raspberry Pi OS bullseye siehe [2].

3.1 GPIO des Raspberry Pi für Pulsweitenmodulation einsetzen [3]

Auf dem Raspberry Pi funktioniert immer noch für Lazarus [4] WiringPi [5] (GPIO Funktionen). Lazarus ist im Raspbian-Repository enthalten (und WiringPi siehe [5]). Die Aktualisierung bei neuen HW-Releases dürfte damit sichergestellt sein.
Ein bereits installiertes Paket wird geprüft mit:

```
gpio –v
```

Die verfügbare Version prüfen:
Testen:

```
gpio -v
gpio readall
```

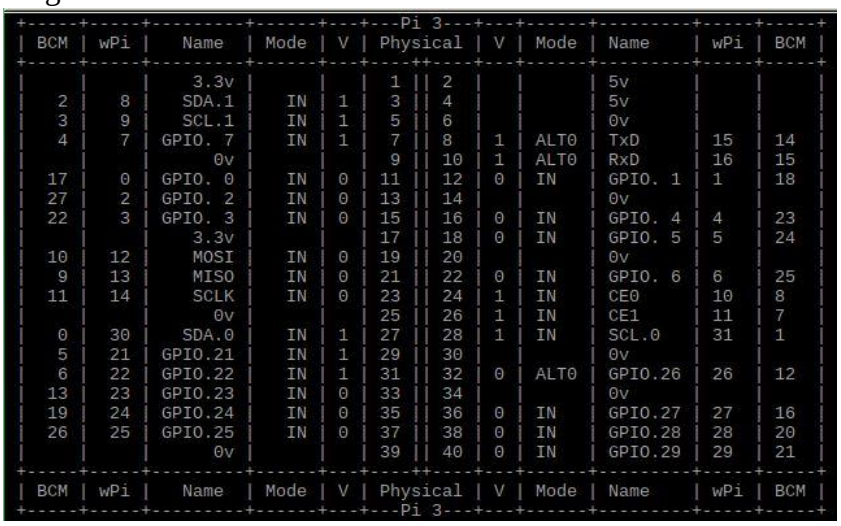
wiringPi.pas
Der Pfad zur Unit muss für den Compiler auffindbar sein. Deshalb in Lazarus einstellen unter *„Projekt→ Projekteinstellungen→ Compilereinstellungen→ Pfade→ Andere Units→“ und hinzufügen „ …/myLib/wiringPi“*.
Einige Funktionen arbeiten mit Threads.

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 (beziehungsweise 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

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. 3: Screenshot nach Befehl „gpio readall“

Pin Modes

Für das Setzen und Lesen einzelner GPIO muss vor einer IO-procedure der Pin-Mode definiert werden.

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

Folgende Modi sind möglich:

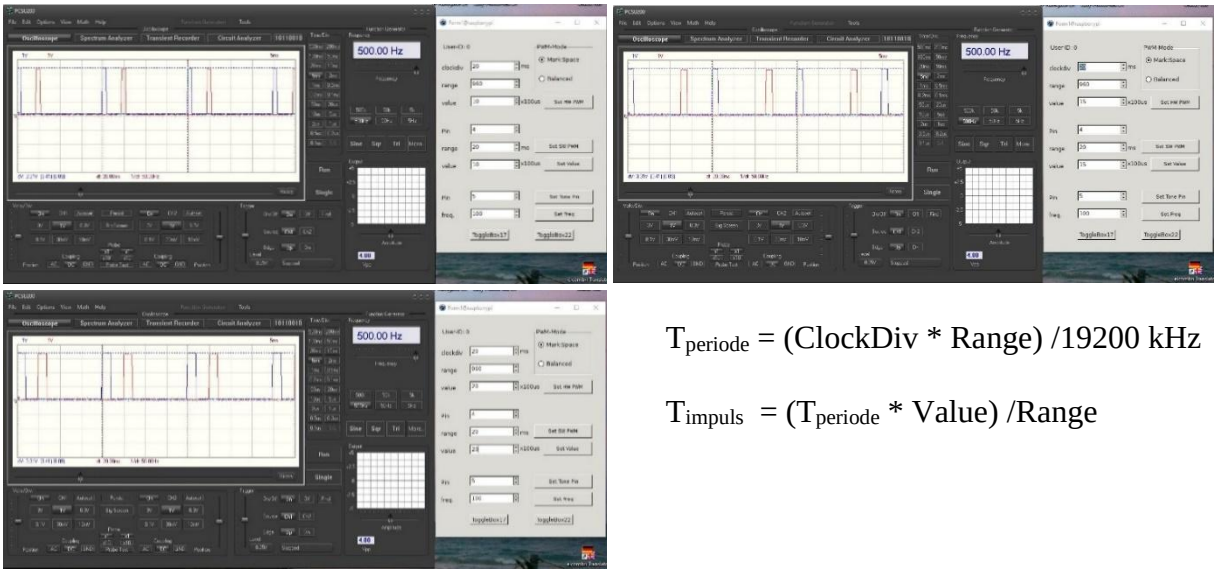
- 0: pmINPUT
- 1: pmOUTPUT
- 2: pmPWM_OUTPUT //nur als root zulässig
- 3: pmGPIO_CLOCK //nur als root zulässig
- 4: pmSOFT_PWM_OUTPUT
- 5: pmSOFT_TONE_OUTPUT
- 6: pmPWM_TONE_OUTPUT

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 (entsprechend RC Modellsteuerung):
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. 4: 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 den Programmen). Das Package „weblaz“ muss installiert sein [6]
(Bei Interesse am kompletten Programmpaket bitte eMail.)

1. Schritt: Datei>Neu>HTTP server Application auswählen. Es wird „program httpproject1“ und „unit Unit1“ angelegt. Dazu aus fpWeb die Componente FPHttpServer platzieren und im Objektinspektor konfigurieren (siehe Abb. 6).
2. Schritt: Datei>Neu>Unit es wird eine „unit Unit2“ hinzugefügt.
3. Schritt: alles speichern (z.B. als WebSev1.lpr, web1.pas, steu1.pas), weiteres wird automatisch gespeichert.

```
program WebSev1;            // Datei>Neu>HTTP  server Application erzeugen (wie in Abb.  5).
                            // (zusätzlich bei neueren lazarus Versionen Web Modul auswählen)
{$mode objfpc}{$H+}

uses
  fphttpapp, Web1, Steu1;

begin
  Application.Title:='WebSev1';
  Application.Port:=8080;
  Application.Threaded:=True;
  Application.CreateForm(TFPWebModule1, FPWebModule1);   // Ist keine richtiges Form, d.h. wird nicht gestartet.)
  Application.Legacyrouting := True;
  Application.Initialize;
  Application.Run;
end.
```

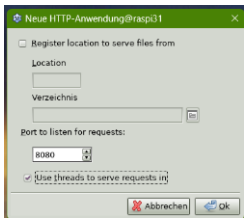


Abb. 5: Modulkonfiguration

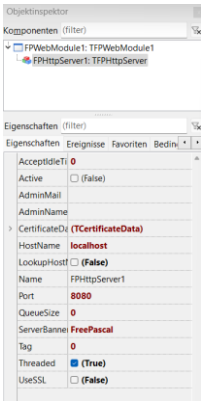


Abb. 6: Objektinspektor

```
unit Web1;                  // Wird bei „HTTP  server Application“ mit erzeugt.
{$mode objfpc}{$H+}
interface
```

```
uses
  cthreads, SysUtils, Classes, httpdefs, fpHTTP, fpWeb, fphttpserver, Steu1;   // cthreads muss bei Linux an erster Stelle stehen

type

  { TFPWebModule1 }

  TFPWebModule1 = class(TFPWebModule)
    FPHttpServer1: TFPHttpServer;                    // automatisch erzeugt
    procedure DataModuleCreate(Sender: TObject);     // zum Start des Moduls          (im Objektinspektor Ereignisse anlegen)
    procedure FPHttpServer1Request(Sender: TObject;  // Gesendete Daten empfangen  (im Objektinspektor Ereignisse anlegen)
      var ARequest: TFPHTTPConnectionRequest;
      var AResponse: TFPHTTPConnectionResponse);
    procedure Send;                                  // selbst anlegen
  private
  public
  end;

var
  FPWebModule1: TFPWebModule1;                       // automatisch erzeugt
  LS1: String;
  LP1: String;
  kF1: String;
  kR1: String;
  LS2: Integer;                                      // selbst anlegen
  LP2: Integer;
  kF2: Integer;
  kR2: Integer;
  Be1: String;

implementation

{$R *.lfm}

{ TFPWebModule1 }

procedure TFPWebModule1.DataModuleCreate(Sender: TObject);
begin
    FPHttpServer1.Active:= true;                     // selbst eintragen
end;

procedure TFPWebModule1.FPHttpServer1Request(Sender: TObject;
  var ARequest: TFPHTTPConnectionRequest;
  var AResponse: TFPHTTPConnectionResponse);
begin
   kR1:= ARequest.ContentFields.Values['myRange1'];
   kF1:= ARequest.ContentFields.Values['myRange2'];
   LS1:= ARequest.ContentFields.Values['myRange3'];
   LP1:= ARequest.ContentFields.Values['myRange4'];
   Be1:= ARequest.ContentFields.Values['myRange5'];
   LS2:= StrToInt(LS1);                              // selbst eintragen
   LP2:= StrToInt(LP1);
   kF2:= StrToInt(kF1);
   kR2:= StrToInt(kR1);
   AResponse.Content := kF1+' '+kR1+' '+LS1+' '+LP1+' '+Be1;
   Send;
end;

  procedure TFPWebModule1.Send;
  begin
    Steuerung.Fahrt(kF2, kR2);
    Steuerung.Licht(LS2, LP2);
    if Be1 = '1' Then                                // selbst eintragen
     begin
       Steuerung.Ende(Be1);
     end;
  end;

initialization
  RegisterHTTPModule('TFPWebModule1', TFPWebModule1);        // automatisch erzeugt
end.
```

```
unit Steu1;                    // Unit mit Datei>Neu>Unit hinzugefügt. Es wird keine unit Form benötigt.

{$mode objfpc}{$H+}

interface

uses
  cthreads, Classes, SysUtils, wiringpi, Process;     // cthreads muss bei Linux an erster Stelle stehen

type
   { Bedienung TSteuerung }
```

```
 TSteuerung = class(TObject)
    procedure Fahrt(kF: Integer; kR: Integer);     //Ruder und Geschwindigleit einstellen
    Procedure Licht(LS: Integer; LP: Integer);     //Spot- und Positionslicht ein/aus          } // selbst eintragen
    procedure Ende(be: String);                    //Befehl vom Web
  private
  public

  end;

var
  Steuerung: TSteuerung;                           // automatisch erzeugt
  TimpF:  Array[0..8] of Integer;                  // selbst anlegen
  s: String;                                       // selbst anlegen

implementation

{ TSteuerung }

procedure TSteuerung.Fahrt(kF: Integer; kR: Integer);  //Ruder und Geschwindigleit einstellen        selbst anlegen
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
    //Verändern TimpSW der SW-PWM
    //impSW in 100 us d.h. 1...2 ms (siehe Array TimpF)
  softPwmWrite(4, TimpF[kR]);     //(Pin,TimpSW) mit TimpSW = TimpF[kR]
end;

Procedure TSteuerung.Licht(LS: Integer; LP: Integer);    //Licht einstellen         selbst anlegen
begin
    //Licht Spot ein/aus
  if LS = 0 then digitalWrite(3,levLOW) else digitalWrite(3,levHIGH);
    //Licht Spot ein/aus
  if LP = 0 then digitalWrite(0,levLOW) else digitalWrite(0,levHIGH);
end;

procedure TSteuerung.Ende(be: String);   //Befehl vom Web zu shutdown einstellen         selbst anlegen
begin
    //TimpHW:= aus
  pwmWrite(1, 0);
    //TimpSW:= aus
  softPwmWrite(4, 0);
    //Licht aus
  digitalWrite(0,levLOW);
  digitalWrite(3,levLOW);
    //Programm beenden
  if be = '1' then
  RunCommand('/bin/bash',['-c', 'sudo /sbin/shutdown -P now'], s);
  writeln(s);
  Halt;
end;

begin
     //Hauptprogramm: Array für Zeiten
  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

    //Setup in WiringPi-Numbering
  wiringPiSetup();
    //Initialisierung HW-PWM WPi 1  Pin 12 und Voreinstellung auf "Halt" entspticht 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
     //ClockDiv und Range so zu wählen, dass Timp genau einstellbar
  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.
```

Zum Beenden nach Anmeldung am RasPi (z.B. zum Bearbeiten des Programms):
Liste aller laufenden Prozesse zum Erkunden der PID von WebSev1 und rpi_camera_web_streaming.py:

$ ps aux

Hintergrund Programme löschen (Zahlen durch PID Ersetzen):

$ sudo kill 547
$ sudo kill 548

3.1.4 Webseiten

Webseite: Steuerung.html (Screenshot siehe Abb. 1)

```
<!DOCTYPE html>
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
  <link rel="stylesheet" href='./static/style.css'/>
  <title>Steuerung</title>
</head>
<body>
<main>
<form name="form1" action="http://<WiFi IP-AdresseRaspi>:8080" target="frame1" method="post" autocomplete="off">
<table style="text-align: left; width: 100%; bottom: 0; border-width: 0;" cellpadding="0" cellspacing="0">
<tbody>
 <tr>
  <td style="height: 8em; width: 0%;"> </td>
  <td style="height: 8em; width: 0%;">
   <div class="slidecontainer2">
    <input type="range" min="0" max="1" value="0" class="slider5" id="myRange5" name="myRange5">
    <span class="sw"><output id="Zeig5">Run</output></span>
   </div>
  </td>
  <td style="height: 8em; width: 0%;"><iframe name="frame1" style="height: 3em; width: 100%; border-width: 0;"
border="0"></iframe></td>
 </tr>
 <tr>
  <td style="vertical-align: top; font-size: 8px; width: 1%;">
  </td>
  <td style="vertical-align: top; font-size: 8px; width: 3%;"><br>
    <big><output id="Zeig1"></output></big>
   <div class="slidecontainer">
    <input type="range" min="0" max="8" value="4" class="slider" id="myRange1" name="myRange1">
   </div>
  </td>
  <td style="vertical-align: top; width: 23%;">
  <img src="r.png" width="45" height="370">
  </td>
  <td style="vertical-align: bottom; width: 30%;"><br>
   <div class="slidecontainer1">
    <input type="range" min="0" max="1" value="0" class="slider1" id="myRange3" name="myRange3">
     <label>Pos-Licht</label> <output id="Zeig3">Aus</output>
   </div>
  </td>
  <td style="vertical-align: bottom; width: 30%;">
   <div class="slidecontainer1">
    <input type="range" min="0" max="1" value="0" class="slider1" id="myRange4" name="myRange4">
     <label>Spot-Licht</label> <output id="Zeig4">Aus</output>
   </div>
  </td>
  <td style="vertical-align: top; width: 3.5%;">
  <img src="v.png" width="45" height="370">
  </td>
  <td style="vertical-align: top; font-size: 8px; width: 3%;"><br>
    <big><output id="Zeig2"></output></big>
   <div class="slidecontainer">
    <input type="range" min="0" max="8" value="4" class="slider" id="myRange2" name="myRange2">
   </div>
  </td>
 </tr>
</tbody>
```

```
</table>
 </form>
</main>
  <script src="./static/skript.js"></script>
</body>
</html>
```

CSS style.css

```
body {
  background-image: url("http://<WiFi IP-AdresseRaspi>:8081/stream.mjpg");
  background-repeat: no-repeat;
  background-position: center top;
  font-family: Arial;
  font-size: 17px;
}
main {
  background: rgba(0, 0, 0, 0);
  color: #101010;
  width: 100%;
}
.slidecontainer {
  transform:
  rotate(90deg)
  translate(-10px, 0px);
  width: 70px;
}
.slider {
  -webkit-appearance: none;
  width: 370px;
  height: 40px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}
.slider:hover {
  opacity: 1;
}
.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 40px;
  height: 40px;
  background: #04AA6D;
  cursor: pointer;
}
.slider::-moz-range-thumb {
  width: 40px;
  height: 40px;
  background: #04AA6D;
  cursor: pointer;
}
.slidecontainer1 {
  width: 100%;
}

.slider1 {
  -webkit-appearance: none;
  width: 60px;
  height: 25px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}
.slider1:hover {
  opacity: 1;
}
.slider1::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
  background: #04AA6D;
  cursor: pointer;
```

```
}
.slider1::-moz-range-thumb {
  width: 25px;
  height: 25px;
  background: #04AA6D;
  cursor: pointer;
}
.sw {color:#000000; font-family:arial,helvetica,sans-serif; font-size:10px;}
.slidecontainer2 {
  width: 100%;
}
.slider5 {
  -webkit-appearance: none;
  width: 22px;
  height: 12px;
  background: #d3d3d3;
  border-radius: 5px;
  outline: none;
  opacity: 0.5;
  -webkit-transition: .2s;
  transition: opacity .2s;
}
.slider5:hover {
  opacity: 1;
}
.slider5::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 10px;
  height: 10px;
  background: #04AA6D;
  cursor: pointer;
}
.slider5::-moz-range-thumb {
  width: 10px;
  height: 10px;
  background: #04AA6D;
  cursor: pointer;
}
```

Javasktipt skript.js

```
var slider1 = document.getElementById("myRange1");
var output1 = document.getElementById("Zeig1");
output1.innerHTML = slider1.value;
slider1.oninput = function() {output1.innerHTML = this.value; document.form1.submit();};

var slider2 = document.getElementById("myRange2");
var output2 = document.getElementById("Zeig2");
output2.innerHTML = slider2.value;
slider2.oninput = function() {output2.innerHTML = this.value; document.form1.submit();};

var slider3 = document.getElementById("myRange3");
var output3 = document.getElementById("Zeig3");
slider3.oninput = Beschriftung1;

function Beschriftung1() {
 if (slider3.value == 0) {
                output3.innerHTML = "Aus";
                } else {
                output3.innerHTML = "Ein";
                };
document.form1.submit();
};

var slider4 = document.getElementById("myRange4");
var output4 = document.getElementById("Zeig4");
slider4.oninput = Beschriftung2;

function Beschriftung2() {
 if (slider4.value == 0) {
                output4.innerHTML = "Aus";
                } else {
                output4.innerHTML = "Ein";
                };
document.form1.submit();
};

var slider5 = document.getElementById("myRange5");
var output5 = document.getElementById("Zeig5");
slider5.oninput = Beschriftung5;
```

```
function Beschriftung5() {
 if (slider5.value == 0) {
                output5.innerHTML = "Run";
                } else {
                output5.innerHTML = "Stop";
                document.form1.submit();};
};
```

3.2 Kameralivestream

Es kann „rpi_camera_web_streaming.py“ genutzt werden. Das reagiert sogar deutlich schneller als raspivid.

rpi_camera_web_streaming.py [7]

```
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()
```

3.3 WiFi Accesspoint einrichten [8]

Raspberry Pi 3 hat ein WLAN zusätzlich ist ein USB WLAN-Stick (über der Wasserlinie) 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 im Home zu Anmeldung und Konfiguratio"
    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.

4 Start der Programme

Zum automatischem Start beim booten wird in die Datei /etc/rc.local folgendes vor exit 0 und iptables-restore eingefügt.

Befehl „sudo nano /etc/rc.local“

```
# zusätzlich       wegen wiringpi müssen root-Rechte bestehen
export USER='root'
export LOGNAME='root'
```

```
# startet die Kommandos Steuerung nach 60 Sekunden im Hintergrund (durch &)
(/bin/sleep 60 && sudo <gesamtes Verzeichnis>/ WebSev1)&
(/bin/sleep 60 && python3 <gesamtes Verzeichnis>/ rpi_camera_web_streaming.py)&
```

Die Webseite mit CSS- und JavaScript-Dateien kommen in das Doc-Verzeichnis des Lighttpd. Nach Anmeldung beim Accesspoint des Raspi kann die Webseite mit http://<WiFi IP-Adresse des RasPi>/<Name der Webseite> aufgerufen werden. Alles ohne jede Anmeldung am RasPi bis zum Shutdown. (Wenn auf dem Bediengerät vorhanden reicht es, die Datei <Name der Webseite> im Browser zu öffnen.)
Der Accesspoint hat natürlich keine Internetverbindung (ohne die Hotspots auf Smartphone, Tablet oder Laptop nicht starten würden).

5 Literaturverzeichnis

[1] E. Boeck, „Seite von Erich Boeck,“ [Online]. Available: http://www.erich-boeck.de/.cm4all/uproc.php/0/imported/BootRaspPi3_ferngesteuert5.pdf?cdp=a&_=1904fad26ab. [Zugriff am 25 06 2024].

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

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

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

[5] github.com, „github.com,“ [Online]. Available: https://github.com/WiringPi/WiringPi/releases/download/2.61-1/wiringpi-2.61-1-armhf.deb. [Zugriff am 25 06 2024].

[6] „fpWeb Tutorial,“ [Online]. Available: https://wiki.freepascal.org/fpWeb_Tutorial. [Zugriff am 28 06 2024].

[7] 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].

[8] P. Schnabel, „Raspberry Pi als WLAN-Router einrichten,“ [Online]. Available: https://www.elektronik-kompendium.de/sites/raspberry-pi/2002171.htm. [Zugriff am 25 06 2024].