

picamctl | Camera Steuerung für Raspberry Pi

Inhaltsverzeichnis

Einleitung	
Programm picamctl	3
Hardware	
Die verwendeten Pins	4
Platine PI_CAMERA_J8 für Header J8:	5
Platine PI_CAMERA_VERTEILER:	6
Konfiguration	
Basiskonfiguration: picamctl_a.conf	7
Weitere Konfigurationsdateien	8
Optionen für libcamera	8
Steuerprogramme	8
Programmstart	
Start ohne Fernbedienung	9
Start mit Fernbedienung	9
Startoptionen	9
USB-Datenträger einbinden	9
Dialoge	
Hauptmenu	10
Setup Programm	10
Pins einstellen/testen:	11
Test Cameraeinstellungen	12
Test Camera	12
Test Bilder	13
Test Filme	14
Run	
Run Loop-Steuerung	15
Run Anzeige fix	16
Run Anzeige fortlaufend	17
Fernbedienung	
PC Rsync Dateien auf PC übertragen	19
vlc Videoplayer	20
Steuerung	
Steuerdateien	21
Beispiele für Steuerdateien	22
Bilder im Zeitraffer aufnehmen:	22
Bilder mit Bewegungssensor und IR-Leds:	22
Bilder an bestimmten Wochentagen:	23
Aufnahme fortlaufend:	23
Aufnahmen zu bestimmten Zeitpunkten	23
Fehlersuche	
Infos Programm	24
Infos Laufzeit	25
Screen-Sitzungen	26
Automatischer Programmstart	
Überwachung mit screenstart	27
Optionen für screenstart	27
Linux 'crontab' einstellen	28
Logdatei	
Logdatei schreiben	28
Logdatei auswerten	28
Installation	
Kurzanleitung	29
Startlink anlegen	29
Screenstart	29
Screenstart compilieren	29
Programmcode	
Tasten Funktionen	30
Beispiel Sensorevent	30
Beispiel Loop	31
Projektdateien	32
Raspberry Pi Boot-Einstellungen	
Camera Konfiguration in /boot/config.txt	32
Zusammenbau der Hardware	
Netzteile einbauen	33
Steuerblock anschließen	34
Endmontage	35
Dokus zum Projekt /c	
GNU General Public License	

Einleitung

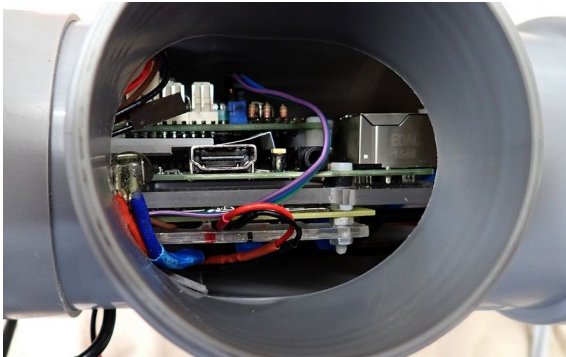
Programm picamctl

Mit Programm 'picamctl' können Cameras mit dem Raspberry Pi gesteuert werden.

- ▷ Steuerung von verschiedenen Cameras
- ▷ Testfunktionen für die verwendeten Pins
- ▷ Automatischer Betrieb mit einer Steuerungsdatei und/oder Bewegungssensor
- ▷ Nachtbetrieb mit Infrarotsensor
- ▷ Einfache Fernbedienung über ssh mit Terminalumschalter 'screen'
- ▷ Bilder und Filme vom Raspberry Pi mit rsync auf den PC übertragen

Die Steuerung passt in ein HT-Rohr mit 75mm Durchmesser. Für Tests ermöglicht diese Ausführung einen einfachen Zugang zum HDMI-Anschluss, Netzwerk und USB-Anschluss über schraubbare Wartungsöffnungen.

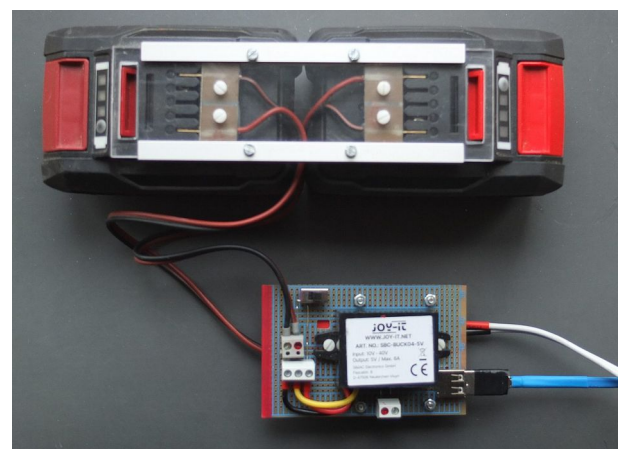
Sie kann, mit dichten Abschlüssen und anderen Leeds versehen, auch in Rohren oder Hohlräumen eingesetzt werden.



Option:

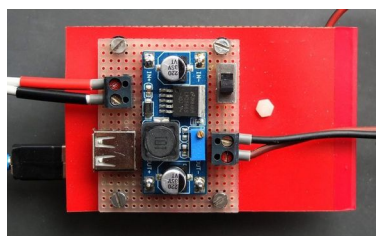
Mobile Stromversorgung mit 2 Akkus vom Typ Eihell PowerX 3.0Ah, 18V.

1. Akku: 5V Spannung für Raspberry Pi
[JOY-IT SBC BUCK4](#)
 DC-DC Step Down Wandler
 In: 10-40V, Out: 5V/6A
 Funktioniert mit Pi sehr gut!



2. Akku: Für Infrarot-Leeds und andere Geräte.
 Leeds eingestellt auf ca. 16V / 300mA

Step Down Regler mit Chip LM2596S
 In: 4.5-35V Out: V Einstellbar/ 2A
 Bringt bei 5V nicht die Leistung für Pi!



Hardware

Die verwendeten Pins

Im Projekt werden immer die physikalischen Pinnummern für J8 verwendet.

Header J8

		J8Nr	J8Nr	
I2C		[1]	[2]	5V
3.3V	← 3.3V	[3]	[4]	5V
SDA	← BCM 2 sda	[5]	[6]	GND
SCL	← BCM 3 scl	[7]	[8]	BCM 14 txd
	BCM 4 gpclk0	[9]	[10]	BCM 15 rdx
GND	← GND	[11]	[12]	BCM 18 pwm0
OUT1	← BCM 17	[13]	[14]	GND
OUT2	← BCM 27	[15]	[16]	BCM 23 ← BSensor
OUT3	← BCM 22	[17]	[18]	BCM 24
	3.3V	[19]	[20]	GND
	BCM 10 misi	[21]	[22]	BCM 25
	BCM 9 miso	[23]	[24]	BCM 8 ce0
	BCM 11 sclk	[25]	[26]	BCM 7 ce1
	GND	[27]	[28]	BCM 1 id_sc
OUT4	← BCM 0 id_sd	[29]	[30]	GND
OUT5	← BCM 5	[31]	[32]	BCM 12 pwm0
	BCM 6	[33]	[34]	GND
IN3	→ BCM 13 pwm1	[35]	[36]	BCM 16
IN2	→ BCM 19 miso	[37]	[38]	BCM 20 mosi ← IN1
	BCM 26	[39]	[40]	BCM 21 sclk
	GND			

OUT1-3 : 500 mA open collector output mit ULN2803A Transistor Array.
Schutzdioden. Erweiterbar auf 8 Pins.

OUT4,5 : Infrarotfilter mit bistabilen Relais ein/aus.

IN1-3 : Input für Tasten usw.

I2C : Für eine Uhr oder andere Sensoren

BSensor: Für Bewegungssensor.

Verwendete J8 Pins:

Pins [11][9] OUT1: IR-Leds für die Nachtausleuchtung. Open Colletor Output.
Für für 300mA beträgt die Spannung ca. 16V

Pins [16][1][9] BSensor: 3.3V Input von Bewegungssensor ANM34111

Pins [27][29] OUT4-5: Infrarotfilter mit bistabilen Relais ein/aus.

Pins [1][3][5][9] I2C: Echtzeituhr PCF8583

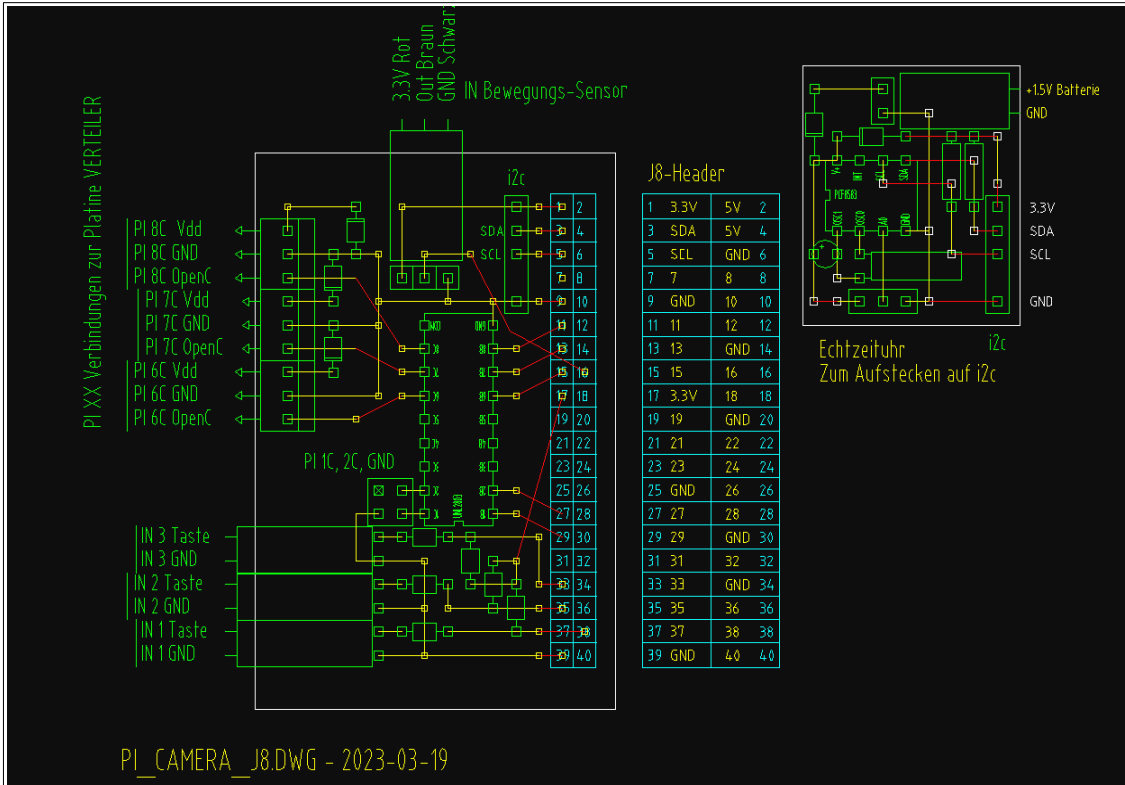
Die verwendeten J8 Pins werden über die Platine [PI_CAMERA_J8](#) verteilt.

Für die 12V Elemente wird die Platine [PI_CAMERA_VERTEILER](#) verwendet.

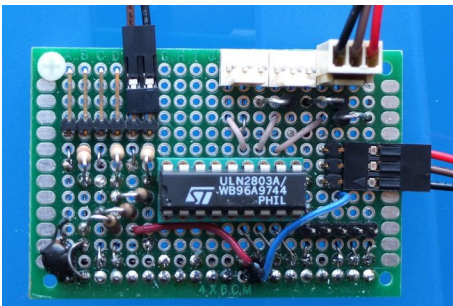
Platine PI CAMERA J8 für Header J8:

Die Platine PI_CAMERA_J8 wird auf den Header J8 aufgesteckt.

- ▷ Stecker für Bewegungssensor: 3,3V, In, GND
- ▷ 3 Stecker für Tasten-Input, 3,3V mit Pullups und Strombegrenzung
- ▷ Treiber ULN_2803 mit 8 Open Collector Outputs 1C bis 8C. Für 12V Ansteuerungen
- ▷ Stecker Pi 8C Verbindung mit Platine PI_CAMERA_VERTTEILER
- ▷ Stecker Pi 7C, N.C.
- ▷ Stecker Pi 6C, N.C.
- ▷ Stecker Pi 1C, 2C, GND für IR-Filter, Verbindung mit Platine PI_CAMERA_VERTTEILER
- ▷ Stecker ic2 für Platine PCF8583 mit Echtzeituhr und Stützbatterie



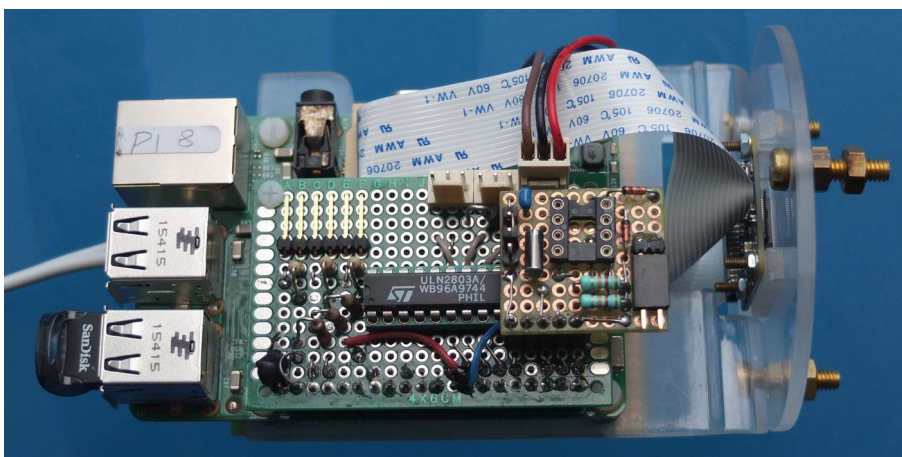
Platine PI_CAMERA_J8 für Pi Header J8:



Platine PCF8583 mit Echtzeituhr:



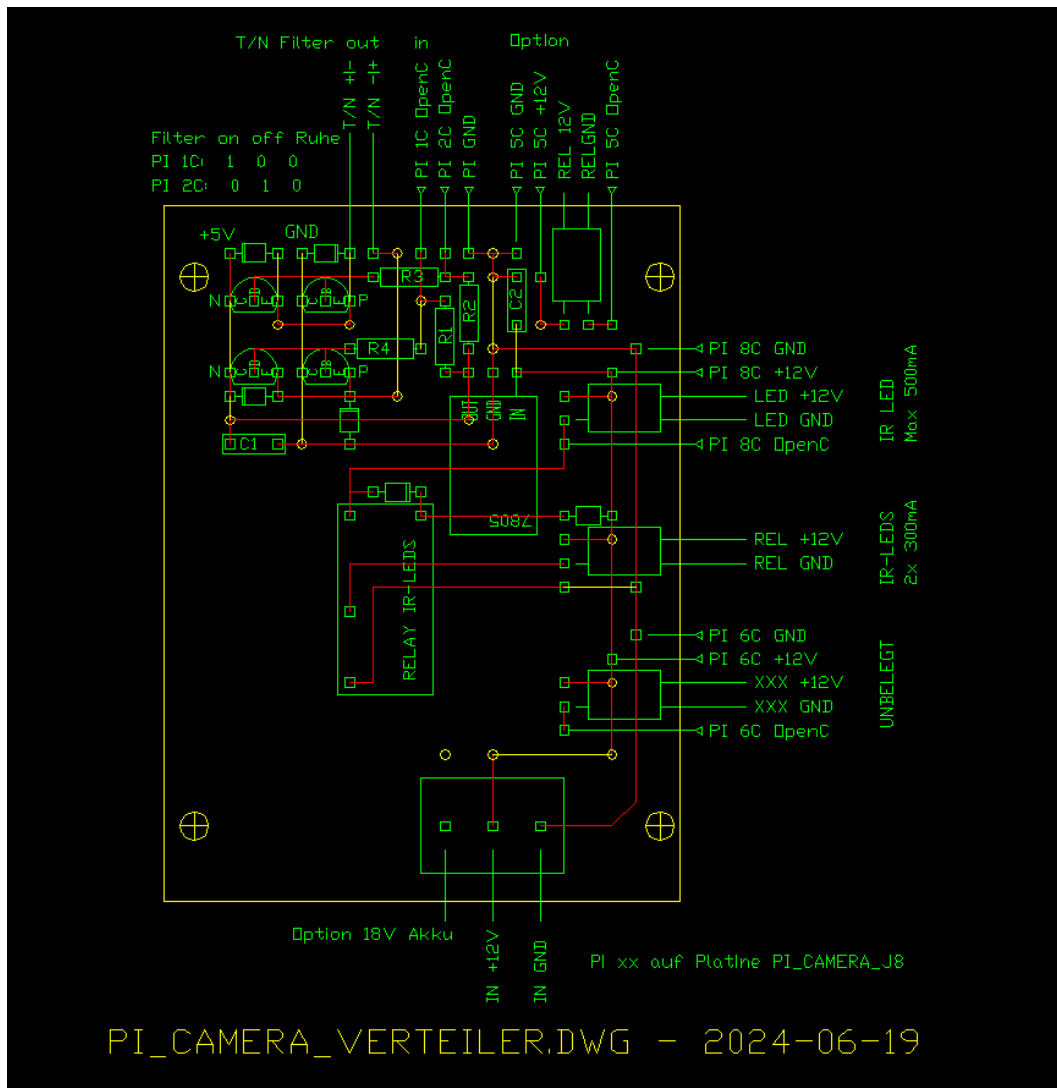
Pi mit Platine PI_CAMERA_J8 und Platine PCF8583 Echtzeituhr:



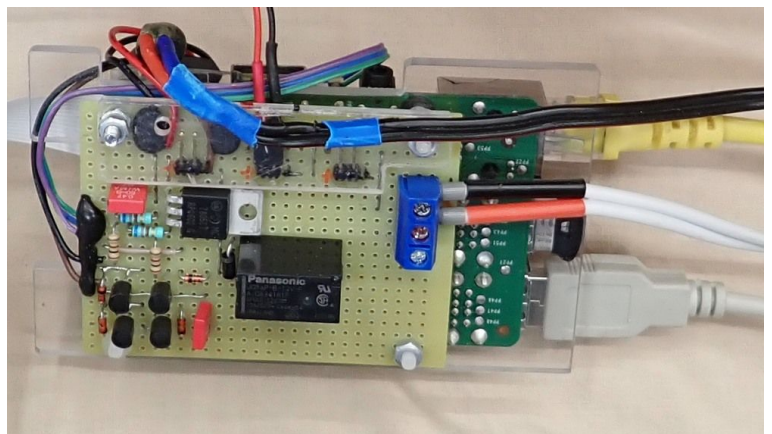
Platine PI CAMERA VERTEILER:

Verteilerplatine PI_CAMERA_VERTEILER für die Open Collector Ausgänge von Platine PI_CAMERA_J8 und Ansteuerung für den IR-Filter.

- ▷ Stecker für 12V Stromversorgung
- ▷ 4 Stecker für die Open Collector Verbindungen mit Platine PI_CAMERA_J8
- ▷ Relay für IR-Leds
- ▷ Verbindungskabel zum Stecker PI_1C,2C,GND auf Platine PI_CAMERA_J8 für den IR-Filter
- ▷ 5V Spannungsregler für IR-Filter Steuerung
- ▷ Transistor-Bücke für IR-Filter Steuerung
- ▷ Verbindungskabel zum IR-Filter



Verteilerplatine PI_CAMERA_VERTEILER
mit Platz für Erweiterungen:



Konfiguration

Basiskonfiguration: picamctl a.conf

Für verschiedene Hardware Zusammenstellungen können verschiedene Basis-Konfigurationen vom Typ 'a'-x' angelegt werden. Mit Option `-c Typ` kann beim Programmaufruf der Typ festgelegt werden. Default ist Typ 'a'.

- ▷ `picamctl_a.conf` Konfiguration für Hardware Typ a, Default
- ▷ `picamctl_b.conf` Konfiguration für Hardware Typ b
- ▷ ...
- ▷ `picamctl_x.conf` Konfiguration für Hardware Typ x

Das Programm sucht beim Start nach der Basiskonfiguration `picamctl_?.conf`:

1. Versuch im Homeordner: `~/config/picamctl/picamctl_?.conf`
2. Versuch im Programmordner: `c/pi/bin/picamctl/bin/_picamctl/picamctl_?.conf`

- Syntax: einfaches C
- '`picamctl.conf.bak`' ist eine Sicherheitskopie von `picamctl_a.conf`

Am Pi sollte die Versuch 1. verwendet werden. Damit wird verhindert, dass bei Programmupdates die Konfiguration versehentlich überschrieben wird.

Beispiel für `picamctl_a.conf`

```
// -----
// picamctl_a: Konfiguration Camerasteuerung Typ 'a'
// Syntax C ähnlich. Siehe c/lib/include/script.h
// Version 0.76 / 2025-03-02
// -----
// Var-Gruppe allgemeine Programinfos für picamctl
VarSetGrpFlt(0,0); // Filter: Gruppe 0 und SubGruppe 0
DatBrowserPi = "mc"; // Datenbrowser für Pi
DatBrowserPc = "caja"; // Datenbrowser für PC
TxtEditor = "nano"; // Texteditor für PC/Pi
LogOn = 1; // Logdatei schreiben
AutoRun = 1; // 1/0 Steuerung automatisch starten

// -----
// Var-Gruppe Infos für picamctl Daten, Pins und Uhr
VarSetGrpFlt(0,1); // Var-Filter: Gruppe 0 und SubGruppe 1

CamInfoStr = "Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter"; // Beschreibung der Konfiguration

CamCtlName = "cam_garten.ctl"; // Steuerdatei ohne Pfad
CamOutDirDef = "~tmp/picamctl/"; // Speicherziel Defaultwert. '/' am Ende
CamOutDir = "~/media/guenther/Scandisk/"; // Tatsächliches Speicherziel. Wird beim Start bestimmt
CamUsbLabel = "Scandisk"; // Option: USB-Label für Speicherziel oder ""
CamLoopShow = 0; // 0/1 Anzeigemodus für Loop

CamUhrName = "PCF8583"; // Option Echtzeituhr
CamUhrSync = "piclock -s"; // Option Echtzeituhr synchronisieren

CamSensorPin = 16; // physikalische J8 Gpio-Pinnummer für Sensor
CamSensorSet = "ir"; // Gpio-Setup für Pin Sensor
CamSensorLab = "Sensor"; // Pin-Bezeichner
CamSensorWait= 2000; // ms Debouncetime

CamIrLedPin = 11; // physikalische J8 Gpio-Pinnummer für IR-Led
CamIrLedSet = "op0"; // Gpio-Setup für Pin IR-Led
CamIrLedLab = "IR-Led"; // Pin-Bezeichner

CamIrFlt1Pin = -1; // physikalische J8 Gpio-Pinnummer für IR-Filter 1
CamIrFlt1Lab = "IR-Flt1"; // Pin-Bezeichner
CamIrFlt2Pin = -1; // physikalische J8 Gpio-Pinnummer für IR-Filter 2
CamIrFlt2Lab = "IR-Flt2"; // Pin-Bezeichner
CamIrFltSet = "op0"; // Gpio-Setup für Pin IR-Led

CamShutterPin= -1; // physikalische J8 Gpio-Pinnummer für Shutter
CamShutterSet= "op"; // Gpio-Setup für Pin Shutter
CamShutterLab= "Shutter"; // Pin-Bezeichner1

// -----
// Var-Gruppe Test Einstellungen für Programm libcamera-xxx
VarSetGrpFlt(0,2); // Var-Filter: Gruppe 0 und SubGruppe 1

CamModul = "imx708";
CamStillName= "camstill.opt"; // Options-Liste für A, ohne Pfad
CamStillOpt = "-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0 --lens-position 0.5 -e jpg --immediate --flush"; // Aktuelle Optionen ohne -o !
CamStillPre = "-t 500000 --saturation 0.0 --lens-position 0.5"; // Optionen für die Bild-Vorschau

CamVidName = "camvid.opt"; // Options-Liste für A, ohne Pfad
CamVidOpt = "-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"; // Aktuelle Optionen ohne -o !
CamVidSec = 5; // Filmlänge in s
CamVidPre = "-t 50000 --saturation 0.0 --height 1080 --width 1920 --lens-position 0.5"; // Optionen für die Film-Vorschau

// -----
// Var-Gruppe Rsync Einstellungen für die Fernsteuerung PC -> Remote-Pi
VarSetGrpFlt(0,3); // Var-Filter: Gruppe 0 und SubGruppe 1

RemoteHost = "pi@pi8w"; // Pi Hostname oder IP-Nummer
PiConfigDir = "/home/pi/.config/picamctl"; // Pi-Ordner für Config-Dateien
RsyncQuelle = "~/media/pi/Scandisk/"; // Quellordner auf Pi
RsyncZiel = "/home/guenther/tmp/picamctl/"; // ZielOrdner auf PC
RsyncWait = 25; // Refresh in Sekunden
BildBrowser = "pix -f"; // PC Bildbetrachter
FilmViewer = "celluloid"; // PC Filmanzeige
TerminalPc = "mate-terminal --tab -x"; // PC Terminalfenster
```

Weitere Konfigurationsdateien

In der Basis-Konfiguration können dann die weiteren Konfigurationsdateien verlinkt werden.

Optionen für libcamera

Befehls-Optionen für Cameratests mit Programm libcamera

- ▷ [camstill.opt](#) Optionsliste für den Befehl [libcamera-still](#)
- ▷ [camvid.opt](#) Optionsliste für den Befehl [libcamera-vid](#)

Beispiel: Optionen aus Liste [camstill.opt](#)

Die eingetragenen Optionen können in der Testfunktion gewählt oder abgewählt werden. In die Liste können weiter Optionen im Format `[#] Option | Kommentar [: Wertebereich]` eingetragen werden.

```
# Option | Kommentar : Wertebereich
-t 1 | Auslösezeit in ms. Belichtungszeit beachten!: ms
#-n | Keine Vorschau
--width 1920 | Bildbreite: 1 bis 3280
--height 1080 | Bildhöhe: 1 bis 2464
#--brightness 1 | Helligkeit: -1.0 to 1.0
#--contrast 1.0 | Kontrast 1.0: 0 bis 32.0
#--saturation 0 | Farbsättigung grau 0.0, normal 1.0: 0.0 bis 32.0
#--sharpness 1.0 | Schärfe 1.0: 0.0 bis 32.0
#--lens-position 6.25 | focus position, 1/m, 100/cm: 0=unendlich bis 32=nah
--list-ctrls
-e jpg | Encoding: jpg, png, rgb, bmp or yuv420
#-s 1 | Capture mit SIGUSR1: 1 bis 0
#--awb auto | AWB mode: auto, incandescent, tungsten, fluorescent, indoor, daylight, cloudy, custom
#--datetime | Filename MMTThhmmss.*
#--timestamp | Filename timestamp
```

Steuerprogramme

Siehe [Steuerung](#)

- ▷ [cam_bild_wh.ctf](#) Bilder zeigesteuert
- ▷ [cam_bild_sensor.ctf](#) Bilder mit Bewegungssensor
- ▷ [cam_Film_wh.ctf](#) Film zeigesteuert
- ▷ [cam_Film_sensor.ctf](#) Film mit Bewegungssensor
- ▷ [cam_carten.ctf](#) Programm für Wildkamera im Garten
- ▷ [cam_*.ctf](#)

Beispiel: Programm für Wildkamera im Garten [cam_carten.ctf](#)
Die Bildauslösung war für Marder und co. zu langsam.

```
// Steuerungsdatei für die Camera-Auslösung mit Programm picamctl
// Format C ähnlich
// DatumZeit: "2023-04-12 12:42:15" oder "2023-04-12" oder "12:42:15"
// Zeitdauer: "12:42:15" oder "42:15" oder ":15"
// EinWochentag: "Mo" "Di" "Mi" "Do" "Fr" "Sa" "So" "-"
// Befehle: von=, bis=, tag=, film=, leds, irfilter, sensor, eshutter
//
// Bilder am Tag
{ rem="Bild: Sensorauslösung ohne IR-Led"
camctl="-t 0 --width 1920 --height 1080 --saturation 0 --lens-position 0.5 -e jpg"
sensor
von="07:01:00" bis="19:00:00"
}
// Filme in der Nacht
{ rem="Film: Sensorauslösung mit IR-Led"
camctl="-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"
film="0:08" // Filmdauer
sensor leds
von="19:01:00" bis="23:59:59"
}
{ rem="Film: Sensorauslösung mit IR-Led"
camctl="-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"
film="0:08" // Filmdauer
sensor leds
von="00:00:01" bis="07:00:00"
}
```


Programmstart

Die Programminstallation wird unter [Installation](#) beschrieben.

Start ohne Fernbedienung

Zum Testen startet man das Programm mit `'picamctl -2'` ohne Fernbedienung, also ohne Terminalumschalter `'screen'`.

```
picamctl -2
```

Beim Start werden dann nebenstehende Systeminformation geprüft/angezeigt.

Wenn die erforderliche Hardware nicht verfügbar ist, wird in den Simulationsmodus geschaltet. Alle Hardwareaufrufe werden dann nur simuliert.

Auf dem PC wird `'picamctl'` immer im Simulationsmodus gestartet.

Anmerkung:

Mit dem symbolischen Link `'~/bin/picamctl'` auf `'picamctl'` kann das Programm auch ohne Pfadangabe gestartet werden.

Dieser Link kann mit `chelp` angelegt werden. Siehe [Startlink anlegen](#).

```

picamctl
printStartDebugInfos: Informationen zur Fehlersuche
User      : guenther
Rechner   : pc780mint
AutorRun: 1 | Option 1 startet die Steuerung automatisch
LogOn    : 1 | Option 1 schreibt Logdatei

Fernbedienung: Terminalumschalter screen
▶ screen: /usr/bin/screen | Ok
Fernbedienung: Startprogramm screenstart
▶ screenstart: /home/guenther/bin/screenstart | Ok
Echtzeituhr: piclock
▶ piclock: /home/guenther/bin/piclock | Ok

Objekt-Log: Daten in Logfile schreiben
LogIsOn  : ON
LogOn    : 1 | Var(LogOn)=1 | Logdatei schreiben
Path     : /media/guenther/Scandisk/20250225_181650.log | Logdatei
fLog     : 0x55611e0761c0 | Filepointer der Logdatei
LogStart : 2025-02-25 18:16:50 | Startzeit
LogEnd   : - | Endzeit

```

Start mit Fernbedienung

Zur Fernbedienung einer Steuerung am Raspberry Pi wird in einem PC-Terminal eine ssh Netzverbindung zum Raspberry Pi hergestellt. Im Terminal kann die Steuerung dann mit dem Befehl `'picamctl [OPTIONS]'` aber **ohne** `[STARTOPT]` am Raspberry-Pi gestartet werden. Sollte die Steuerung bereits laufen, so werden nur die Aus- und Eingaben ins Terminalfenster umgeleitet.

```
picamctl [OPTIONS] # aber ohne [STARTOPT]
```

Wird die laufende Steuerung mit dem Befehl `Exit Terminal` verlassen, so läuft sie am Raspberry-Pi auch Terminalaus und -eingabe weiter. Ein erneuter Aufruf von `'picamctl'` von einem beliebigen PC-Terminal aus leitet die Aus- und Eingaben laufende Steuerung ins Terminal um.

Die notwendigen Befehle zum Umleiten der Aus- und Eingaben erledigen die folgenden zwei Hilfsprogramme:

- ▷ `screen` : Linux Terminalumschalter für Fernzugriffe. Sie werden als Screen-Sitzungen bezeichnet. Installation mit: `sudo apt-get install screen`.
- ▷ `screenstart` : Programm zu einfachen Verwaltung von Screen-Sitzungen. aus Projekt `/c`. Kompilieren mit Programm `chelp`.

Startoptionen

Der Hilfeaufruf von `'picamctl'` erklärt die möglichen Programm- und Startoptionen:

```

picamctl -h
Aufruf: picamctl [OPTIONS] [STARTOPT]
OPTIONS:
-h      Help
-m      mtrace on. Als erste Option wählen!
-c Typ  Camera Typ: a,b ... Default a
-A      Steuerung automatisch starten und merken
-a      Steuerung nicht starten und merken
-d, -D  Debug

Die genaue Beschreibung findet man unter Help!

STARTOPT:
-2 Startmodus ohne Fernsteuerung, ohne 'screen'
-1 Startmodus reserviert für 'screen'

```

[OPTIONS] Diese Programmoptionen werden an das Programm selbst übergeben.
`-m` aktiviert Funktion `mtrace` zur Speicherüberwachung. Infos werden am Programmende angezeigt.
`-A` startet am Raspberry Pi sofort die Steuerung ohne Menu. Am PC wird der 'PC Rsync' Fernbedienungsdialog angezeigt.
`-a` schaltet die OPTION `-A` wieder aus. Beide Optionen setzen das `AutoRun`-Flag in Config.

[STARTOPT] Die Startoption steuert den Programmstart in Funktion `StartCheck()`.
`keine` Die aufgerufene Programm-Instanz wird sofort durch den folgenden neuen Programmaufruf beendet:
`screenstart picamctl [OPTIONS] -1`

Durch diesen Aufruf wird dann entweder eine bereits bestehende `'screen-Sitzung'` mit `'picamctl'` aktiviert oder eine neue `'screen-Sitzung'` mit `'picamctl'` angelegt.
`-1` Wird intern zur Aktivierung/Kontrolle einer `'screen-Sitzung'` mit `'screenstart'` verwendet! Siehe oben!
`-2` Programm zum Testen der Hardware, Software und der Einstellungen ohne Fernbedienung starten. Beim Umschalten des Terminal in `'screen-Sitzung'` können Ausgaben/Meldungen verschluckt werden.

USB-Datenträger einbinden

Das Programm läuft auch ohne X-Windows. Die USB-Datenträger werden dann mit der Projekt `c/` Funktion `mountUSB()` gemounted.

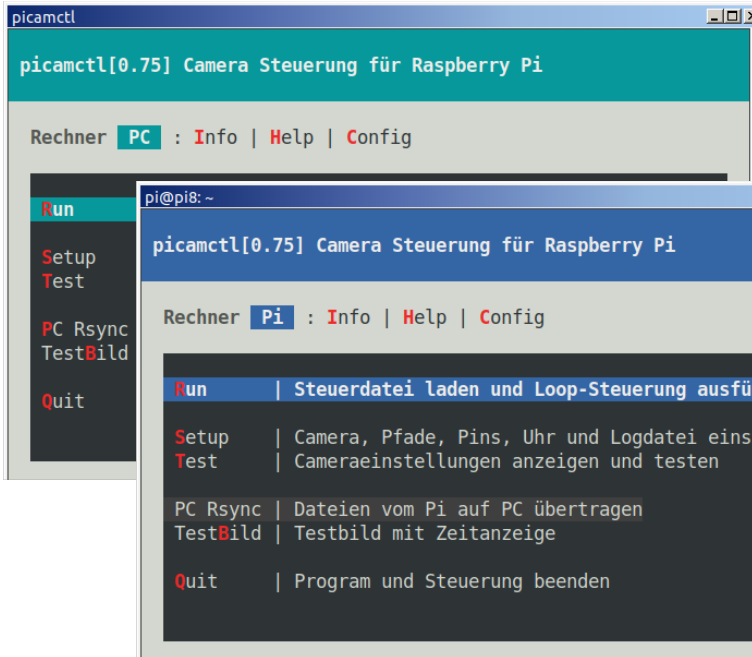
Das Mounten ohne Passwort kann mit dem PolicyKit legitimiert werden. Im Verzeichnis `'c/bin/infosys/bin/_infosys/'` befindet sich eine passende Regeldatei `'10_pi.pkla'`.

Diese Datei muss als root in den Ordner `'/etc/polkit-1/localauthority/50-local.d'` kopiert werden.

Die Zahl 10 kann durch eine andere Zahl ersetzt werden.

Dialoge

Hauptmenu

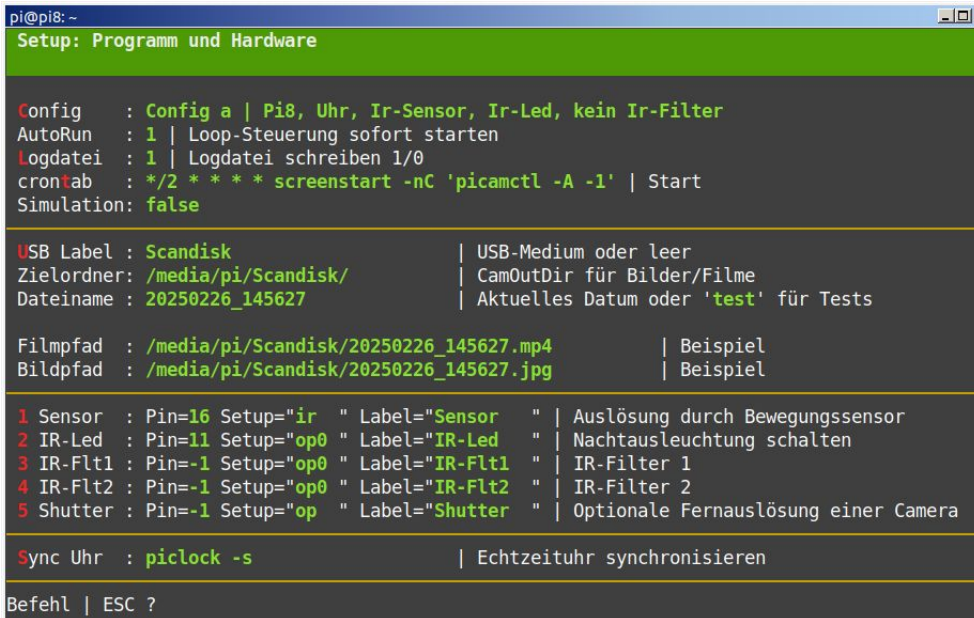


Zur besseren Unterscheidung wird das Hauptmenu am PC und am Rasperry Pi in verschiedenen Farben angezeigt.

- Run** Steuerung starten
- Setup** Programmeinstellungen
- PC sync** nur am PC verfügbar
- TestBild** Testbildfolge für Filme

- Info:** Ausführliche Infos zum Programm: Umgebung, Variablen, Programmpfade, Pins, Rsyn, Infos zu Screen.
- Help:** Die einfache Hilfedatei `1_read.me` anzeigen.
- Config:** Konfigurationsdatei `'picamctl.conf'` des Programms anzeigen.

Setup | Programm



- Camera Typ aus Config
- crontab:** Überprüft alle 2 Minuten ob das Programm läuft.
- USB-Datentäger mit dem angezeigten Label wird automatisch gemountet.
- Beispiele für Pfade
- Pins einstellen und testen
- Option: Uhr mit NTP synchronisieren

- AutoRun:** Wert `1`: Die Steuerung wird ohne Menu mit der eingestellten Steuerdatei gestartet. Der AutoRun-Wert kann mit der Programm-Option `-A` gesetzt und mit `-a` gelöscht werden. Der Wert bleibt in Config gespeichert.
- Logdatei:** `1`: Logdatei schreiben. Der Wert wird in Config gespeichert.
- crontab:** Befehlszeile aus crontab für den automatischen Start. Befehl `t` ruft crontab `-e` zum Einstellen auf. [Siehe automatischer Aufruf mit crontab.](#)
- Simulation:** `true`: Steuerung nur simulieren. Der Wert wird über die Verfügbarkeit von Camera und Software ermittelt.

Pins | einstellen/testen:

Beispiel: Pin 1 Sensor

```

pi@pi8:~$ Pin: Bewegungssensor

3.3V      [ 1][ 2] 5V
BCM 2  sda  [ 3][ 4] 5V
BCM 3  scl  [ 5][ 6] GND
BCM 4  gpclk0 [ 7][ 8] BCM 14 txd
GND     [ 9][10] BCM 15 rdx
BCM 17  [11][12] BCM 18 pwm0
BCM 27  [13][14] GND
BCM 22  [15][16] BCM 23
3.3V   [17][18] BCM 24
BCM 10  misi [19][20] GND
BCM 9   miso [21][22] BCM 25
BCM 11  sclk [23][24] BCM 8  ce0
GND    [25][26] BCM 7  ce1
BCM 0   id_sd [27][28] BCM 1  id_sc
BCM 5   [29][30] GND
BCM 6   [31][32] BCM 12 pwm0
BCM 13  pwm1 [33][34] GND
BCM 19  miso [35][36] BCM 16
BCM 26  [37][38] BCM 20 mosi
GND    [39][40] BCM 21 sclk

          J8Nr└─┬─┘ J8Nr

Pin J8Nr : 16      | Pin [16] Header J8 | 0 für nicht verwenden | GPIO 1-40
Label    : Sensor | Gpio-Pinbezeichner
Setup    : ir      | Gpio-Setupstring für Pin
Wait     : 2000    | ms Debouncetime für Event Pins
Testen   :         | Gpio-Funktion testen

Befehl | RETURN | ESC ?
    
```

Pin J8Nr: Es wird die tatsächliche physikalische Pinnummer vorgeschlagen. 0 für Pin nicht verwenden.
Label: Pinbezeichner für GPIO-Interface
Setup: Setupstring für GPIO-Interface
Wait: Wartezeit in ms nach einem Pin-Event

Optionen für Setupstring für GPIO-Interface

```

pi@pi8:~$ Optionen für Pins: Mehrere Optionen möglich
o Output
i Input
p/n Logik positiv/negativ
d/s Open Drain/Sourcen
0/1 Start- oder Endpegel
r Event 0-1, rising
f Event 1-0, falling

Gpio-Setup Optionen: i r
    
```

Testen: Pin einbinden und testen

```

pi@pi8:~$ Pin: Bewegungssensor

Pin      : 16
Setup    : ir
I/O-Modus: e
Label    : Sensor | BSensor benötigt Zeit zum Stabilisieren!

Call: GpioAddPin(16, Sensor, ir)

GpioAddPin(16,"Sensor","ir")

[16] BCM 23 | Sensor | Setup:ir RF:0x1 fdh:-1 EF:0x1 fde:-1 Start:-1 Typ:C EvTime:0 EvFlag:0
└─Flags:i p 0x0

Gpio Event-Handler setzen | Debounce 2000 ms
GpioReqEventHandle: [23] RF:0x1 EF:0x1 Label:Sensor fgpio=3
[16] BCM 23 | Sensor | Setup:ir RF:0x1 fdh:4 EF:0x1 fde:4 Start:-1 Typ:C EvTime:0 EvFlag:0
└─User:Sensor Flags:i p k 0x1

Event-Pin aktiv | Befehle: Event | Taste | ESC ...
▶ EventHandler | J8Nr:16 | EvTime :4568663

--> Taste: 'taste_select'
Pin:16 | EvTime: 4568663
Befehlsaufruf ...
Time: 4569665 | TimeDiff: 1002
Time: 4570663 Fortsetzung

Event-Pin aktiv | Befehle: Event | Taste | ESC ...
    
```

Ein Testlauf ...
 Pin hinzufügen: Nr, Name, Setup
 Event-Pin noch nicht aktivieren!
 GPIO Rückmeldung
 Event-Handle für Pin aktivieren
 GPIO Rückmeldung
 Loop-Anzeige der Testfunktion ...
 Bewegung hat Event auf Pin 16 zum Zeitpunkt 4568663 ms ausgelöst
 Event-Handler liefert 'taste_select'
 Daten des Events abfragen und gewünschten Befehl aufrufen ...
 Debounce-Time abwarten ...
 Loop-Anzeige der Testfunktion ...

Der Sourcecode für die beschriebene Testfunktion kann im [Codebeispiel](#) eingesehen werden.

Test | Cameraeinstellungen

```

picamctl

Cameraeinstellungen testen

Config : Config a | Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter
Simulation: true

Camera | Infos zur aktuellen Camera
-----|-----
Bilder | Camera Setup und Test
Filme  | Camera Setup und Test
Quit   |

```

Test | Camera

```

pi@7: ~
Infos zur aktuellen Camera

Die verwendeten Foto- und Filmbefehle sind in camlib.h definiert.

Simulation : false

Config      : a | Aktuelle Camera-Konfiguration
Config Path : /home/pi/.config/picamctl/picamctl_a.conf
Config Info : Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter | CamInfoStr

Foto Befehl : 'libcamera-still'
Foto Optionen : '-t 1 --width 1920 --height 1080 -e jpg'
Foto Opt-Liste : 'camstill.opt' | Auswahl von Optionen für Fotos

Film Befehl : 'libcamera-vid'
Film Optionen : '-t 0 -n'
Film Opt-Liste : 'camvid.opt' | Auswahl von Optionen für Filme

Infos zur Camera abfragen

Befehl | ESC ?

```

```

pi@7: ~/c/pi
Infos zur Camera

dtoverlay="imx708" | Kernel-Treiber der Camera. Siehe /boot/config.txt

libcamera-hello --list-cameras | Liste der Cameras
-----|-----
Available cameras
-----|-----
0 : imx708_noir [4608x2592 10-bit RGBG] (/base/soc/i2c0mux/i2c@1/imx708@1a)
   Modes: 'SRGBB10_CSI2P' : 1536x864 [120.13 fps - (768, 432)/3072x1728 crop]
   2304x1296 [56.03 fps - (0, 0)/4608x2592 crop]
   4608x2592 [14.35 fps - (0, 0)/4608x2592 crop]

lsmod | grep --color=always imx708 | Kernel-Treiber
-----|-----
imx708          24576  3
v4l2_fwnode    20480  2 bcm2835_unicam,imx708
v4l2_async     20480  4 bcm2835_unicam,dw9807_vcm,imx708,v4l2_fwnode
videodev      274432 26 bcm2835_unicam,dw9807_vcm,bcm2835_isp,imx708,v4l2_fwnode
mc             53248 14 bcm2835_unicam,dw9807_vcm,bcm2835_isp,imx708,bcm2835_c

cat /boot/firmware/config.txt | grep --color=always camera | Konfiguration
-----|-----
# Automatically load overlays for detected cameras
camera_auto_detect=1

CamInit(NULL)
CamInit() ...
[0:10:57.964483738] [5475] INFO Camera camera_manager.cpp:327 libcamera v0.4.0+53-2915
[0:10:58.155784350] [5478] WARN CameraSensorProperties camera_sensor_properties.cpp:47
[0:10:58.156291349] [5478] WARN CameraSensorProperties camera_sensor_properties.cpp:47
[0:10:58.533002585] [5478] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning - please con
[0:10:58.550907548] [5478] WARN CameraSensor camera_sensor_legacy.cpp:501 'imx708_noir
[0:10:58.559169532] [5478] INFO RPI vc4.cpp:447 Registered camera /base/soc/i2c0mux/i2
[0:10:58.559585531] [5478] INFO RPI pipeline_base.cpp:1121 Using configuration file '/>
Mode selection for 2304:1296:12:P
SRGBB10_CSI2P,1536x864/0 - Score: 3400
SRGBB10_CSI2P,2304x1296/0 - Score: 1000
SRGBB10_CSI2P,4608x2592/0 - Score: 1900
[0:10:58.573954502] [5475] INFO Camera camera.cpp:1202 configuring streams: (0) 2304x1
[0:10:58.576092497] [5478] INFO RPI vc4.cpp:622 Sensor: /base/soc/i2c0mux/i2c@1/imx708

Ende mit q

```

Test | Bilder

Testfunktionen für Bilder:

```

pi@pi8:~
Setup Bilder: Config a | Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter

Optionsliste für Libcamera-still | camstill.opt
# Option | Kommentar : Wertebereich
-t 0 | Auslösezeit in ms. Belichtungszeit beachten!: ms
#-n | Keine Vorschau
--width 1920 | Bildbreite: 1 bis 3280
--height 1080 | Bildhöhe: 1 bis 2464
#--brightness 0.1 | Helligkeit: -1.0 to 1.0
#--contrast 0.5 | Kontrast 1.0: 0 bis 32.0
--saturation 0 | Color saturation, where 1.0 = normal and 0.0=greyscale
#--sharpness 1.0 | Schärfe 1.0: 0.0 bis 32.0
--lens-position 0.5 | focus position, m=1/p, cm=100/p: 0=unendlich bis 32=nah
#--shutter 500 | Set fixed shutter speed
-e jpg | Encoding: jpg, png, rgb, bmp or yuv420
#-s 1 | Capture mit SIGUSR1: 1 bis 0
#--awb auto | AWB mode: auto, incandescent, tungsten, fluorescent, indoor,
--immediate | 1/0 Perform first capture immediately, with no preview phase
--flush | Flush output data as soon as possible
Optionen ohne '#' werden verwendet.
RETURN Gewählte Optionszeile bearbeiten
Optionsliste bearbeiten

HDMI-Vorschau: -t 500000 --saturation 0.0 --lens-position 0.5

Sensor:16 | IR-Leds:11 | IR-Filter:-1,-1-> '?'

Die Tests verwenden die Optionsliste ohne -o
Testbild speichern: /media/pi/Scandisk/test.jpg
BSensor testen : /media/pi/Scandisk/test.jpg

Quit und Optionsliste speichern | ESC Abbruch ohne speichern
  
```

Options-Liste **camstill.opt**

Optionen mit # werden für den Test nicht verwendet.

Option zum Ändern mit Cursortasten auswählen

RETURN wählte Option ändern

Optionsliste bearbeiten

Hdmi-Vorschau am Monitor anzeigen.

Sensor | Leds | IR-Filter

Testbild mit den Optionen aus der Optionsliste speichern

BSensor Bewegungssensor mit Optionen testen

Für die Tests werden alle gewählten Optionen der Optionsliste verwendet. Nur die Speicheroption **-o** wird automatisch mit den Zielpfaden aus dem Setup berechnet.

Für die gewählte Option kann mit **RETURN** der Dialog **Edit Option** angezeigt werden.

```

pi@7:~
Edit Option

Optionen aus Liste: /home/pi/.config/picamctl/camstill.opt

Option: --lens-position | focus position, 1/m, 100/cm
Wert : 6.25 | 0 bis 32
Aktiv : nein | Wert '#' deaktiviert die Option

ESC | Optionswert oder '#' oder ' ' 0.8
  
```

Mit Befehl **Optionsliste bearbeiten** können alle Befehle der Options-Liste **camstill.opt** im Editor bearbeitet werden.

Die Zeilen der Options-Liste haben immer folgendes Format:

[#] Option | Kommentar [: Wertebereich]

```

# Option | Kommentar : Wertebereich
-t 0 | Auslösezeit in ms. Belichtungszeit beachten!: ms
#-n | Keine Vorschau
--width 1920 | Bildbreite: 1 bis 3280
--height 1080 | Bildhöhe: 1 bis 2464
#--brightness 0.1 | Helligkeit: -1.0 to 1.0
#--contrast 0.5 | Kontrast 1.0: 0 bis 32.0
--saturation 0 | Color saturation, where 1.0 = normal and 0.0=greyscale
#--sharpness 1.0 | Schärfe 1.0: 0.0 bis 32.0
--lens-position 0.5 | focus position, m=1/p, cm=100/p: 0=unendlich bis 32=nah
#--shutter 500 | Set fixed shutter speed
-e jpg | Encoding: jpg, png, rgb, bmp or yuv420
#-s 1 | Capture mit SIGUSR1: 1 bis 0
#--awb auto | AWB mode: auto, incandescent, tungsten, fluorescent, indoor, daylight, cloudy, custom
--immediate | 1/0 Perform first capture immediately, with no preview phase
--flush | Flush output data as soon as possible
#--wrap 5 | When writing multiple output files, reset when it reaches this
#--datetime | Filename MMTThhmmss.*
#--timelapse 20 | Time interval (in ms) between timelapse captures
#--timestamp | Filename timestamp
  
```

Test | Filme

Testfunktionen für Filme:

```

pi@pi8:~
Setup Filme: Config a | Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter

Optionsliste für libcamera-vid | camvid.opt
# Option | Kommentar : Wertebereich
-t 20 | Filmlänge in millis: 0 unendlich
-n | 1/0 Kein Vorschaubild
#-f | Vorschau Vollbild
--width 1920 | Bildbreite: 1 bis 3280
--height 1080 | Bildhöhe: 1 bis 2464
--lens-position 0.5 | focus position, m=1/p, cm=100/p: 0=unendlich bis 32=nah
--saturation 0.0 | Color saturation, where 1.0 = normal and 0.0=greyscale
#--save-pts arg | Save a timestamp file with this name
--brightness 0.1 | Helligkeit: -1.0 to 1.0

Optionen ohne '#' werden verwendet. -t und -o werden immer berechnet.
RETURN Gewählte Optionszeile bearbeiten
Optionsliste bearbeiten

HDMI-Vorschau: -t 50000 --saturation 0.0 --height 1080 --width 1920 --lens-position 0.5
Sensor:16 | IR-Leds:11 | IR-Filter:-1,-1-> '?'

Die Tests verwenden die Optionsliste ohne -o und -t
Optionen: "-t 0 -n --width 1920 --height 1080 --lens-position 0.5 --saturation 0.0 --bright
Dauer : 00:00:05 | 5 sec

Testfilm speichern: /media/pi/Scandisk/test.h264
Sensor testen : /media/pi/Scandisk/test.h264

Quit und Optionsliste speichern | ESC Abbruch ohne speichern

```

Options-Liste **camvid.opt**

Filmlänge wird automatisch '-t 0' gesetzt.
Optionen mit # werden für den Test nicht verwendet.
Zum Ändern mit Cursortasten auswählen

RETURN wählte Option ändern
Optionsliste bearbeiten

Hdmi-Vorschau am Monitor anzeigen.
Vorschau-Optionen aus Setup
Sensor | Leds | IR-Filter

Dauer des Films. Immer Option **-t 0** verwenden!

Testfilm mit den Optionen aus Optionsliste speichern
Sensor Bewegungssensor mit Optionen testen

Für die Tests werden alle gewählten Optionen der Optionsliste verwendet. Nur die Speicheroption **-o** wird automatisch mit den Zielpfaden aus dem Setup berechnet. Es wird immer Option **-t 0** verwendet! Die Filmlänge wird vom Programm gesteuert.

Für die gewählte Option kann mit **RETURN** der Dialog **Edit Option** angezeigt werden.

```

pi@7:~
Edit Option

Optionen aus Liste: /home/pi/.config/picamctl/camstill.opt

Option: --lens-position | focus position, 1/m, 100/cm
Wert : 6.25 | 0 bis 32
Aktiv : nein | Wert '#' deaktiviert die Option

ESC | Optionswert oder '#' oder ' ' 0.8

```

Mit Befehl **Optionsliste bearbeiten** können alle Befehle der Options-Liste **camvid.opt** im Editor bearbeitet werden.

Die Zeilen der Options-Liste haben immer folgendes Format: [#] Option | Kommentar [: Wertebereich]

```

# Option | Kommentar : Wertebereich
-t 20 | Filmlänge in millis: 0 unendlich
-n | 1/0 Kein Vorschaubild
#-f | Vorschau Vollbild
--width 1920 | Bildbreite: 1 bis 3280
--height 1080 | Bildhöhe: 1 bis 2464
--lens-position 0.5 | focus position, m=1/p, cm=100/p: 0=unendlich bis 32=nah
--saturation 0.0 | Color saturation, where 1.0 = normal and 0.0=greyscale
#--save-pts arg | Save a timestamp file with this name

```

Run

Beim Aufruf des Befehls `picamctl` am am Raspberry-Pi gibt es folgende Möglichkeiten:

1. Programm `picamctl` läuft bereits in einer Screen-Sitzung:
Im Terminal wird die zuletzt gewählte Anzeige der laufenden Sitzung angezeigt.
2. Programm `picamctl` läuft nicht:
Es wird eine Screen-Sitzung mit `picamctl` angelegt.
AutoRun=1: Aus der configurierten Steuerdatei wird die aktuelle Liste der Tagesbefehle wird erzeugt und die [Steuerung](#) sofort gestartet.
AutoRun=0: Es wird der Dialog Loop-Steuerung zur Auswahl einer Steuerdatei angezeigt.
3. Programm `picamctl` läuft im Testmodus:
Es wird keine weitere Instanz von `picamctl` gestartet.

Run | Loop-Steuerung

Die Steuerung der Camera wird durch eine [Steuerdatei](#) beschrieben. Im Dialog [Loop-Steuerung](#) kann die gewünschte Steuerdatei gewählt, angezeigt, bearbeitet und/oder simuliert werden. Mit dem Befehl `Run` wird die Steuerung danach gestartet.

Beim Start der [Loop-Steuerung](#) wird die gewählte Steuerdatei geparkt und eine aktuelle Liste der erforderlichen Tagesbefehle erstellt. Die Liste enthält die Befehle vom Startzeitpunkt bis Mitternacht. Um Mitternacht wird eine neue Tagesliste für den nächsten Tag erstellt.

```

pi@pi8: ~
Loop-Steuerung: Steuerdatei wählen, Loop starten

Config: Config a | Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter
Terminalmultiplexer: aktiv
Simulation           : false

Steuerdatei: /home/pi/.config/picamctl/cam_garten.ctl
Anzeigen
Bearbeiten

Tagesbefehle | Steuerdatei parsen und Tagesbefehle erzeugen/anzeigen
Simulation   | Tagesbefehle nur simulieren
Run          | Tagesbefehle erzeugen und Loop-Steuerung starten
Ende        | Steuerung und Programm beenden
Quit        | Hauptmenu aufrufen

Befehl | ESC ?

```

Camera Typ A
Terminalmultiplexer: aktiv/nein
Simulation: true/false

Steuerdatei wählen, anzeigen oder bearbeiten
Die Steuerdatei wird im Kapitel [Steuerung](#) beschrieben.

Tagesbefehle aus Steuerdatei ermitteln
Simulation: Simulation starten
Run: Steuerung starten

- Tagesbefehle:** Aus den Definitionen der Steuerdatei wird die Liste der Tagesbefehle erstellt und geprüft.
Simulation: Ausführung der Tagesbefehle simulieren.
Run: Loop-Steuerung: Tagesbefehle erstellen. Tagesbefehle fortlaufend durchführen. Bei aktivem Terminalumschalter kann dann das Terminal mit `Exit Terminal` verlassen werden, ohne die Steuerung zu beenden.
Ende: Steuerung und Programm beenden.

Beispiel: Steuerungsdatei `cam_garden.ctl`

```

/ Steuerungsdatei für die Camera-Auslösung mit Programm picamctl
// Format C ähnlich
// DatumZeit: "2023-04-12 12:42:15" oder "2023-04-12" oder "12:42:15"
// Zeitdauer: "12:42:15" oder "42:15" oder ":15"
// EinWochentag: "Mo" "Di" "Mi" "Do" "Fr" "Sa" "So" "-"
// Befehle: von=, bis=, tag=, film=, leds, irfilter, sensor, eshutter
//
// Bilder am Tag
{ rem="Bild: Sensorauslösung ohne IR-Led"
camctl="-t 0 --width 1920 --height 1080 --saturation 0 --lens-position 0.5 -e jpg"
sensor
von="07:01:00" bis="19:00:00"
}
// Filme in der Nacht
{ rem="Film: Sensorauslösung mit IR-Led"
camctl="-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"
film="0:08" // Filmdauer ohne weitere Sensorauslösung
sensor leds
von="19:01:00" bis="23:59:59"
}
{ rem="Film: Sensorauslösung mit IR-Led"
camctl="-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"
film="0:08" // Filmdauer ohne weitere Sensorauslösung
sensor leds
von="00:00:01" bis="07:00:00"
}

```

Mit dem Befehl **Tagesbefehle** wird aus der Steuerungsdatei die Liste der Tagesbefehle ab dem aktuellen Zeitpunkt erzeugt. Dabei werden die Daten geprüft und es werden die benötigten Hardwareeinstellungen ermittelt. Das Ergebnis ist eine kompakte Befehlsliste. Jede Zeile besteht aus einer Befehlsfolge einem Zeitpunkt und einem Infostring.

Aus der obigen Steuerdatei wird für den Zeitpunkt 2025-02-26 18:39:09 die folgende Tagesliste erzeugt:

```

picamctl
Tagesbefehle ab 2025-02-26 18:39:09

Datei: /home/guenther/c/pi/bin/picamctl/bin/_picamctl/cam_garten.ctl
Verwendete Pins: 16=Sensor 11=IR-Led

Loop-Befehle:
b Bild aufnehmen
F f Film starten/beenden
s Bewegungssensor ein
l IR-Leds ein
i IR-Filter ein
c Camerabefehl
# Remark
+ Init
- Blockende
n Tagesbefehle neu berechnen
* Rückgabe: Befehl erledigt
! Rückgabe: Fehler

Nr| Bef | Datetime | Info
0 # 2025-02-26 18:39:09 Bild: Sensorauslösung ohne IR-Led
1 +cbs 2025-02-26 18:39:09 -t 0 --width 1920 --height 1080 --saturation 0 --lens-po
2 bs 2025-02-26 18:39:09
3 - 2025-02-26 19:00:00
4 # 2025-02-26 19:01:00 Film: Sensorauslösung mit IR-Led
5 +cFsl 2025-02-26 19:01:00 -t 0 -n -q 100 --width 1920 --height 1080 --saturation 0
6 Fs 2025-02-26 19:01:00
7 fs 2025-02-26 19:01:08
8 - 2025-02-26 23:59:59
9 n 2025-02-27 00:00:00

Ende mit q
  
```

Startzeit der Tagesliste

Die benötigte Hardware

Liste der möglichen Tagesbefehle

Befehlsbeispiel: **+cFsl**
 + Init
 c Camera
 F Film
 s Sensor on
 IR-Leds verwenden

Tagesbefehle ab 2025-02-26 18:39:09

0 Kommentar
 1 Init Befehlsblock: **+cFsl**
 2 Film mit Sensor
 3
 4 Blockende
 5 Neuer Tag. Tagesbefehle neu berechnen

Erklärungen:

- 0 # Kommentarzeile
- 1 **+cbs** Befehlsblock initialisieren:
 Camera, Bilder, Sensor verwenden, Camera-Befehl "-t 0 --width 1920 --height 1080 --saturation 0 --lens-position 0.5 -e jpg"
- 2 **bs** Blockanfang 2025-02-26 18:39:09. Bilder durch Sensorauslösung speichern.
- 3 - Blockende um 2025-02-26 19:00:0
- 4 # Kommentar zum nächsten Block
- 5 **+cFsl** Befehlsblock initialisieren:
 Camera, Film, Sensor verwenden, IR-Leds verwenden, Camera-Befehl "-t 0 --width 1920 --height 1080 --lens-position 0.8 ..."
- 6 **Fs** Blockanfang 2025-02-26 19:01:00. Filmaufnahme mit Sensorauslösung speichern
- 7 **fs** Filmdauer ohne weitere Sensorauslösung ist 8 Sekunden. fs_2025-02-26 19:01:08 - Fs_2025-02-26 19:01:00 = 8
- 8 - Blockende um 2025-02-23 23:59:59
- 9 **n** Tagesbefehle um 2025-02-27 00:00:00 neu berechnen

Run | Anzeige fix

Im Terminal kann zwischen einer ausführlichen **fortlaufenden Anzeige** und einer **fixen Blockanzeige** gewählt werden.

```

pi@pi8:~
Steuerung 2025-02-26 18:36:20

0 # 2025-02-26 18:28:11 Bild: Sensorauslösung ohne IR-Led
1 +cbs 2025-02-26 18:28:11 -t 1 --width 1920 --height 1080 --saturat
2 bs 2025-02-26 18:28:11
3 - 2025-02-26 19:00:00
4 # 2025-02-26 19:01:00 Film: Sensorauslösung mit IR-Led
5 +cFsl 2025-02-26 19:01:00 -t 0 -n -q 100 --width 1920 --height 1080
6 Fs 2025-02-26 19:01:00
7 fs 2025-02-26 19:01:08
8 - 2025-02-26 23:59:59
9 n 2025-02-27 00:00:00

Exit Terminal | Stopp | Anzeige | Infos
  
```

Aktuelle Zeit

Erledigte Befehle

Aktueller Befehl

Die nächsten Befehle

Nächster Tag. Tagesbefehle neu berechnen

- Exit Terminal:** Bei aktiver Fernsteuerung wird das Terminal verlassen und das Programm läuft im Hintergrund ohne Terminal weiter.
- Stopp:** Steuerung beenden. Hauptmenu anzeigen.
- Anzeige:** Anzeige zwischen **fix** und **fortlaufend** umschalten und neu aufbauen.
 Bei Verwendung des Terminalumschalters kann die Anzeige gestört werden.
- Infos:** Mögliche Tages-Befehle, Loop-Status anzeigen und alle Befehlstasten anzeigen

Run | Anzeige fortlaufend

Die fortlaufende Anzeige zeigt ausführlich den Ablauf der verschiedenen Funktionsaufrufe an.

```

pi@pi8: ~
2025-02-26 19:44:24|Befehl:+cFsl* 2025-02-26 19:44:23|SenBef:-|FilmOn:false|Anzeige|Info
getTagBef: NxtI:2 > Fs 2025-02-26 19:44:23
Tag:- 2025-02-26 23:59:59 NxtI:5 Bef:s 00:00:00 Sen:F 00:00:08
LoopExecCtl: Befehl:s 00:00:00 SenBef:F 00:00:08
Sensor starten | SenBef:F
GpioReqEventHandle: [23] RF:0x1 EF:0x1 Label:Sensor fgpio=4
[16] BCM 23 | Sensor | Setup:ir RF:0x1 fdh:6 EF:0x1 fde:6 Start:-1 Typ:C EvTime:0 EvFla
g:0
└─GPIO23User:Sensor Flags:i p k 0x1
LoopExecCtl > Befehl:s* SenBef:F TagBef:- FilmOn:false

2025-02-26 19:44:46|Befehl:s* 00:00:00|SenBef:F|FilmOn:false|Anzeige|Info

Loop-Befehle:
b Bild aufnehmen
F f Film starten/beenden
s Bewegungssensor ein
l IR-Leds ein
i IR-Filter ein
c Camerabefehl
# Remark
+ Init
- Blockende
n Tagesbefehle neu berechnen
* Rückgabe: Befehl erledigt
! Rückgabe: Fehler

Loop-Status:
Time : 2025-02-26 19:44:46 | Zeit
Befehl : s* 00:00:00 | Loop-Befehl
SenBef : Fs 00:00:08 | Sensorbefehl
TagBef : - 2025-02-26 23:59:59 | Tagesbefehl
TagNxtI: 5 | Index für nächsten Tagesbefehl
FilmOn : false | Filmaufnahme läuft
WaitSec: 1 | Wartezeit für Taste
LogIsOn: 1 | Logdatei in Verwendung
Debug : 0 | Debugmodus

Exit Terminal|Stopp|Anzeige|Debug|Prog-Check|Log|Test

2025-02-26 19:45:07|Befehl:s* 00:00:00|SenBef:F|FilmOn:false|Anzeige|Info
    
```

Beispiel: Befehl: **+cFsl***, * steht für erledigt
Nächster Tagesbefehl **Fs**

Befehl **s** und SenBef **F** ausführen
Diese Befehl aktiviert den Sensor durch starten des GPIO Eventhandlers.

Statusanzeige mit aktueller Zeit

Ausgabe des Befehls **Info**
Liste der möglichen Tagesbefehle

Statusinformationen von Loop

Befehl	Aktueller Befehl, * für erledigt
SenBef	Ausführung nach Sensor-Event
TagBef:	Nächster Tagesbefehl

Vollständige Befehlsliste

- Anzeige** Anzeige zwischen **fix** und **fortlaufend** umschalten und neu aufbauen.
- Debug** Zwischen Debug 0 und 1 umschalten. Debug 1 zeigt auch übersprungene Befehle an.
- Prog-Check** Zeigt mit **screenstart -i 'picamctl** Details zur Programminstanz und zur Screen-Sitzung an.
- Log** Details zu Logdatei.
- Test** Löst manuell ein Sensor-Event aus. Falls kein Sensor aktiv ist, wird ein Bild aufgenommen.

Das folgende Fenster zeigt den Ablauf einer durch einen Sensor ausgelösten Filmaufnahme an:

```

Taste: taste_select | SenBef:F
Film starten | Dauer:8sec ...
LoopExecCtl: Befehl:F 00:00:00 SenBef:F 00:00:08
CamExecCtl('Fs')
CamCallVid()
IRLed=1
libcamera-vid -t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8 --brigh
tness 0.1 -o /media/pi/Scandisk/20250226_212209.h264 &
Filmeinde setzen | Sensor neu starten
[29:20:49.009465337] [17202] INFO Camera camera_manager.cpp:299 libcamera v0.0.4+22-923f5d70
[29:20:49.179213483] [17203] INFO RPI raspberrypi.cpp:1476 Registered camera /base/soc/i2c0mux/i2c@
1/imx708@1a to Unicam device /dev/media3 and ISP device /dev/media0
[29:20:49.180752640] [17202] INFO Camera camera.cpp:1028 configuring streams: (0) 1920x1080-YUV420
[29:20:49.181354406] [17203] INFO RPI raspberrypi.cpp:851 Sensor: /base/soc/i2c0mux/i2c@1/imx708@1a
- Selected sensor format: 2304x1296-SBGGR10 1X10 - Selected unicam format: 2304x1296-pBAA
LoopExecCtl > Befehl:fs SenBef:F TagBef:- FilmOn:true

2025-02-26 21:22:09|Befehl:fs 2025-02-26 21:22:17|SenBef:F|FilmOn:true |Anzeige|Info

> EventHandler | J8Nr:16 | EvTimeMs :1332158
Taste: taste_select | SenBef:F
Film läuft 8 sec bis fs 2025-02-26 21:22:17 | Sensor neu starten
2025-02-26 21:22:14|Befehl:fs 2025-02-26 21:22:17|SenBef:F|FilmOn:true |Anzeige|Info

> EventHandler | J8Nr:16 | EvTimeMs :1335146
Taste: taste_select | SenBef:F
Film läuft 8 sec bis fs 2025-02-26 21:22:22 | Sensor neu starten
2025-02-26 21:22:18|Befehl:fs 2025-02-26 21:22:22|SenBef:F|FilmOn:true |Anzeige|Info

> EventHandler | J8Nr:16 | EvTimeMs :1338663
Taste: taste_select | SenBef:F
Film läuft 8 sec bis fs 2025-02-26 21:22:26 | Sensor neu starten
2025-02-26 21:22:25|Befehl:fs 2025-02-26 21:22:26|SenBef:F|FilmOn:true |Anzeige|Info
LoopExecCtl: Befehl:fs 2025-02-26 21:22:26 SenBef:F 00:00:08
CamExecCtl('fs')
CamKillFilm() kill(17202,SIGKILL)
IRLed=0
Filmeinde
LoopExecCtl > Befehl:fs* SenBef:F TagBef:- FilmOn:false
    
```

Bei Aktivierung des Sensors liefert der GPIO Eventhandler die Taste **taste_select**.
Dieser Tastenbefehl startet den Film mit **Fs**

IR-Leds einschalten
Camerabefehl aus der Steuerdatei. Wird mit **&** im Hintergrund aufrufen.

Ausgaben von libcamera-vid

Befehl **fs** würde den Film nach 8 sec stoppen

Ein weiteres Sensor-Event verlängert um 8 sec.

Weiteres Sensor-Event an **Pin 16**. Der nächste Sensorstart nach einem Delay von **EvTimeMS**.

Filmzeit 8 sec ist ohne Sensor-Event abgelaufen

Filmbefehl mit **kill** beenden
IR-Leds ausschalten

Run: Anzeige fix

```

pi@pi8: ~
Steuerung 2025-02-26 18:36:20
0 #      2025-02-26 18:28:11  Bild: Sensorauslösung ohne IR-Led
1 +cbs   2025-02-26 18:28:11  -t 1 --width 1920 --height 1080 --saturat
2 bs     2025-02-26 18:28:11
3 -      2025-02-26 19:00:00
4 #      2025-02-26 19:01:00  Film: Sensorauslösung mit IR-Led
5 +cFsl  2025-02-26 19:01:00  -t 0 -n -q 100 --width 1920 --height 1080
6 Fs     2025-02-26 19:01:00
7 fs     2025-02-26 19:01:08
8 -      2025-02-26 23:59:59
9 n      2025-02-27 00:00:00

Exit Terminal | Stopp | Anzeige | Infos

```

Aktuelle Zeit
Erledigte Befehle
Aktueller Befehl
Die nächsten Befehle
Nächster Tag. Tagesbefehle neu berechnen

- Exit Terminal:** Bei aktiver Fernsteuerung wird das Terminal verlassen und das Programm läuft im Hintergrund ohne Terminal weiter.
- Stopp:** Steuerung beenden. Hauptmenu anzeigen.
- Anzeige:** Anzeige zwischen `fix` und `fortlaufend` umschalten und neu aufbauen. Bei Verwendung des Terminalumschalters kann die Anzeige gestört werden.
- Infos:** Mögliche Tages-Befehle, Loop-Status anzeigen und alle Befehlstasten anzeigen

Fernbedienung

PC Rsync | Dateien auf PC übertragen

Die Funktion **PC Rsync** ist nur auf dem PC aktiviert. Sie dient dazu, die aufgenommenen Bilder und Filme vom Raspberry Pi auf den PC zu übertragen. Zum Betrachten können die gewünschten Anzeige-Programme eingestellt werden.

Beim Start von **picamctl** am PC mit Option **-A** (AutoRun) erscheint sofort der **PC Rsync** Dialog. Am PC werden Überschriften in Cyan angezeigt. Falls sich Daten auf einem USB-Medium befinden, muss dieses gemountet sein. Wenn **picamctl** am Raspberry Pi nicht läuft, kann können die Daten durch einen einmaligen Aufruf von **picamctl** am Raspberry Pi gemountet werden.

```

picamctl
PC Rsync: Bilder und Filme vom Remote-Pi auf den PC übertragen
Remote-Pi steuern und Dateien bearbeiten

Mit Befehl 'Rsync PI ► PC' können Bilder/Filme vom Remote-Pi forlaufend
auf den PC übertragen werden. Nach 'Wait' Sekunden wird die Übertragung
wiederholt. Die Funktionen benötigen eine ssh-Verbindung zum Remote-Pi.

Einstellungen am PC:

Remote Host Pi : pi@pi8w | Benutzer@Host oder IP-Nummer
Pi Config-Dir : /home/pi/.config/picamctl | Configdateien am Remote-Pi
Rsync Pi-Quelle: /media/pi/Scandisk/ | Datenquelle am Remote-Pi
Rsync PC-Ziel : /home/guenther/tmp/picamctl/ | Zielordner am PC
Wait : 25 | RSync Wartezeit in Sekunden

1 Befehl Bildanzeige : 'pix -f'
2 Befehl Filmanzeige : 'celluloid'
3 Befehl Dateianzeige: 'caja'
4 Befehl Terminal : 'mate-terminal --tab -x'

PC-Ziel : Rsync PI ► PC | Bilder anzeigen | Filme anzeigen
Remote-Pi : Datenquelle | Config-Dir | SSH Verbindung
Menu : ESC
    
```

Das Programm arbeitet unabhängig vom Rasperry Pi.

Remote Host Pi SSH Verbindung zum Rasperry-Pi
Pi Config-Dir Ordner der Configdateien am Remote-Pi
Rsync Pi-Quelle Ordner mit Bildern/Filmen und Logdatei
Rsync PC Ziel Rsync Zielordner am PC
Wait Rsync nach **Wait** Sec erneut aufrufen.

Anzeigeprogramme am PC
1 Befehl Bildanzeige Programm für Bilder
2 Befehl Filmanzeige Programm für Filme
3 Befehl Dateianzeige Dateibrowser
4 Befehl Terminal Terminal

Rsync PI > PC Synchronisiert den Ordner Pi-Quelle mit PC-Ziel
Bilder anzeigen Bilder aus PC-Ziel anzeigen.
Filme anzeigen Filme aus PC-Ziel anzeigen.
Datenquelle Ordner Pi-Quelle mit Dateibrowser anzeigen
Config-Dir Config Ordner vom Rasperry-Pi mit Dateibrowser anzeigen
SSH Verbindung Terminal mit SSH Verbindung zum Rasperry-Pi aufbauen

Beispiel: Ablauf von **Rsync PI > PC**

```

guenther@pc780mint: ~
Quelle und Ziel alle 25 Sekunden synchronisieren:
Befehl: rsync -av 'pi@pi8:/media/pi/picamctl/' '/home/pi/tmp/picamctl/'

Taste für Rsync now | ESC Rsync Beenden
rsync -av 'pi@pi8:/media/pi/picamctl/' '/home/pi/tmp/picamctl/'
receiving incremental file list
./
20230407_142816.jpg
20230407_142846.jpg

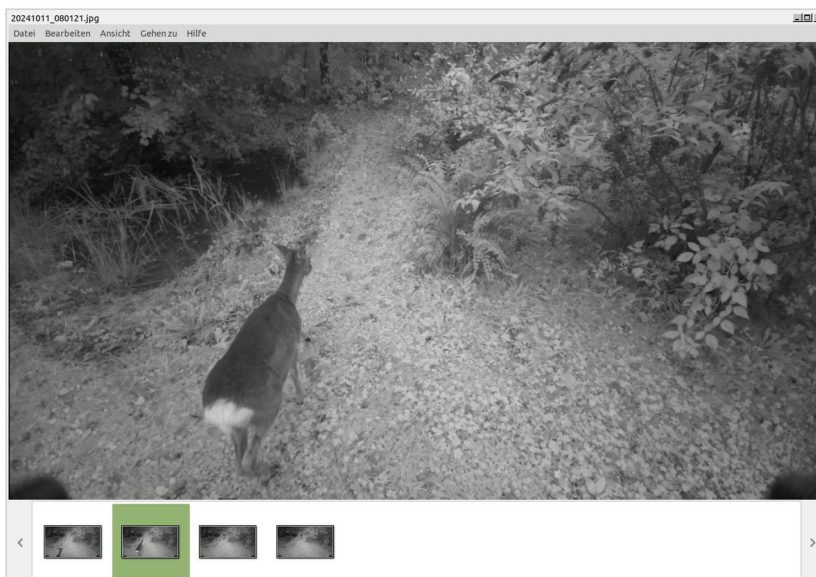
sent 66 bytes received 8,848,053 bytes 3,539,247.60 bytes/sec
total size is 8,845,651 speedup is 1.00
    
```

Eine beliebige Taste ruft Rsync sofort auf. Rsync Befehl. ESC beendet Rsync.

Es wurden zwei neue Bilder übertragen

Rsync wird nach **Wait** Sekunden wiederholt.

Beispiel: **Bilder anzeigen**



Die übertragenen Bilder können im gewählten Bildbetrachter angezeigt werden. Die Vorschaubilder werden dabei vom Bildbetrachter automatisch aktualisiert.

Aufnahme mit Camera Module 2 NoIR

vlc | Videoplayer

Für den vlc Videoplayer gibt es zusätzliche Funktionen.

Befehl: **Filme anzeigen**

```

picamctl
Filme anzeigen

Programm: vlc
PC-Ziel : /home/guenther/tmp/picamctl/
Suffix  : *.h264
Hilfe   : /home/guenther/c/pi/bin/picamctl/bin/_picamctl/vlc_help.txt

vlc läuft bereits: kill -9 3719

Film wählen
20250228_230054_Fuchs.h264
20250303_011554_Marder.h264
20250303_021027.h264
20250304_005437_Katze.h264
20250304_211022_Dachs.h264
20250304_220420_Dachs.h264
20250304_220611.h264
20250305_010021.h264
20250305_205958.h264

```

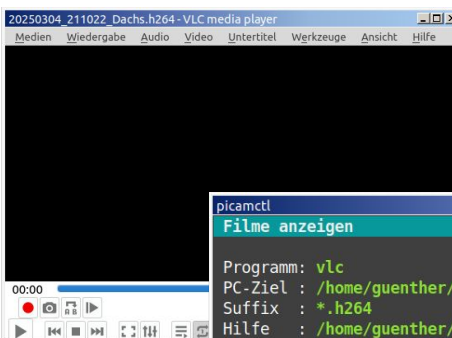
Liste der Suffixe aus `runpc.c`
Funktionstasten für vlc

Alle laufenden vlc Instanzen werden gekillt

Filmwahl

Liste der Videos aus PC-Ziel

Auswahlcursor



Filmanzeige mit vlc

```

picamctl
Filme anzeigen

Programm: vlc
PC-Ziel : /home/guenther/tmp/picamctl/
Suffix  : *.h264
Hilfe   : /home/guenther/c/pi/bin/picamctl/bin/_picamctl/vlc_help.txt

Filmpath: '/home/guenther/tmp/picamctl/20250304_211022_Dachs.h264'
Befehl  : vlc '/home/guenther/tmp/picamctl/20250304_211022_Dachs.h264' 2>/dev/

Tasten für vlc:
+,[    schneller
-.,]   langsamer
=      Geschwindigkeit normal
E      Next Frame
shift+s Snapshot
strg+e Effekte einstellen
strg+q vlc beenden

Befehl: Film wählen | vlc Einstellungen

```

Hilfe für vlc aus Datei `vlc_help.txt`

Befehl: **Vlc Einstellungen**

Günstige Einstellungen für Nachtfilme

```

picamctl
Einstellungen für vlc

Befehl: cat ~/.config/vlc/vlcrc

507 contrast=0.980000
510 brightness=0.930000
513 hue=4.000000
519 gamma=1.280000
538 sharpen-sigma=1.040000
3016 sout-x264-preset=ultrafast
3019 sout-x264-tune=film
3728 qt-system-tray=0
3776 qt-privacy-ask=0
3982 snapshot-path=/home/guenther/tmp/picamctl
3994 snapshot-sequential=1
4054 vout=any
4057 video-filter=sharpen
4243 input-record-path=/home/guenther/tmp/picamctl
4345 random=1
4351 repeat=1

Weiter mit Taste

```

Steuerung

Für automatische Bild- und Filmaufnahmen können Befehlsfolgen in einer [Steuerdatei](#) definiert werden. Diese Datei kann im Dialog [Steuerung](#) ausgewählt, angezeigt und bearbeitet werden.

Die Camera-Befehle werden dabei durch einen oder mehrere [Steuerblöcke](#) beschrieben. Das Format ist einfaches C.

Steuerdateien

Elemente der Steuerungsdatei:

```
// Kommentar      | Kommentare ab //
/* ... */         | Kommentarblöcke werden übersprungen
{ ... }          | ein Steuerblock
```

Steuerblöcke können folgende [Befehle](#) und [Parameter](#) enthalten:

von	= "DatumZeit"	Start Datum und/oder Tageszeit
bis	= "DatumZeit"	Ende Datum und/oder Tageszeit
tag	= "EinWochentag"	Liste von Wochentagen. Die Steuerung erfolgt nur an diesen Tagen
wh	= "Zeitdauer"	Wiederholung nach Zeitdauer oder Pause nach Filmen
bild		Bildmodus, Default
film	= "Zeitdauer"	Filmmodus. Aufnahmedauer des Films in "hh:mm:ss"
sensor		Bewegungsmelder verwenden
leds		IR-Leds für Nachtaufnahmen verwenden
irfilter		IR-Filter einschalten
camctl	= "BefehleCamera"	Befehlsstring für die Camera
rem	= "Infostring"	Infos zum Befehlsblock
eshutter		Auslöser externer Camera ansteuern

Beispiele zum Format der Parameter:

DatumZeit	"2023-04-12 12:42:15" oder "2023-04-12" oder "12:42:15"
Zeitdauer	"02:42:15" oder "42:15" oder "05"
EinWochentag	"Mo" "Di" "Mi" "Do" "Fr" "Sa" "So" "-"
BefehleCamera	"-t 200 -n -q 100 -w 1920 -h 1080"
Infostring	"Nachtaufnahmen mit Pi-Camera"

Aus diesen Befehlen können ein oder mehrere Steuerblöcke zusammengestellt werden. Aus diese Angaben werden dann die aktuellen Tages-Befehle compiliert. Jeweils um 0 Uhr werden neue Tages-Befehle berechnet.

Die Steuerblöcke müssen zeitlich aufsteigend sortiert und ohne zeitliche Überlappungen definiert werden. Beim Berechnen der Tages-Befehle werden die Daten geprüft.

Beispiel einer gültigen Steuerungsdatei:

```
// Steuerungsdatei für die Camera-Auslösung mit Programm picamctl
// Format C ähnlich
// DatumZeit: "2023-04-12 12:42:15" oder "2023-04-12" oder "12:42:15"
// Zeitdauer: "12:42:15" oder "42:15" oder ":15"
// EinWochentag: "Mo" "Di" "Mi" "Do" "Fr" "Sa" "So" "- "
// Befehle: von, bis, tag, bild, film, sensor, led, irfilter, camctl, rem, eshutter
//
// Auslösung: Fotos, von 7-18 Uhr im Zeitraum 2023-01-02 bis 2023-04-07, alle 1:30 Minuten
{ von="2023-01-02 07:00:00" // Startdatum, täglich von 07:00:00
  bis="2023-04-07 18:00:00" // Enddatum, täglich bis 07:00:00
  wh="1:30" // Wiederholung alle 1:30 Minuten
}

// Auslösung: Fotos, nur Montags 19-24 Uhr, mit Bewegungsmelder, IR-Leds
{ von="19:00:00"
  bis="23:59:59"
  sensor
  led
  tag="Mo"
}

/* gültiger Block, auskommentiert
// Filmaufnahme zu einem fixe Zeitpunkt
{ von="2023-01-02 07:00:00"; bis="20:00:00" led
  Film="1:30" // Film Dauer 1:30 Minuten
}
*/
```

Beispiele für Steuerdateien

Bilder im Zeitraffer aufnehmen:

```
// Bildtest1: Fotos, Auslösung alle 0:20 Sekunden
{ rem="Foto alle 30 Sekunden" camctl="-t 5 -n -w 1920 -h 1080" wh="30:00" leds }
```

```
picamctl
Tagesbefehle ab 2025-03-02 10:20:59
Datei: /home/guenther/c/pi/bin/picamctl/bin/_picamctl/cam_test.ctl
Verwendete Pins: 11=IR-Led

Loop-Befehle:
b Bild aufnehmen
F f Film starten/beenden
s Bewegungssensor ein
l IR-Leds ein
i IR-Filter ein
c Camerabefehl
# Remark
+ Init
- Blockende
n Tagesbefehle neu berechnen
* Rückgabe: Befehl erledigt
! Rückgabe: Fehler

Nr| Bef | Datetime | Info
0 # 2025-03-02 10:20:59 Foto alle 30 Sekunden
1 +cbl 2025-03-02 10:20:59 -t 5 -n -w 1920 -h 1080
2 b 2025-03-02 10:20:59
3 b 2025-03-02 10:50:59
4 b 2025-03-02 11:20:59
5 b 2025-03-02 11:50:59
```

Befehl: **Tagesbefehle** anzeigen
Datei mit den Definitionen
Es wird Pin 11 verwendet

Liste der möglichen Tagesbefehle

Compilierte Liste der Tagesbefehle

Befehl: **#** Infotext anzeigen
Befehle: **+cbl** Blockanfang | IR-LedsCam-| Befehlsstring
Befehl: **b** Bild aufnehmen | Zeitpunkt

Bilder mit Bewegungssensor und IR-Leds:

```
{ sensor leds camctl="-t 5 -n -q 100 -w 1920 -h 1080" rem="Test2" }
```

```
picamctl
Steuerung 2025-03-02 10:33:29
0 # 2025-03-02 10:33:17 Test2
1 +cbsl 2025-03-02 10:33:17 -t 5 -n -q 100 -w 1920 -h 1080
2 bs 2025-03-02 10:33:17
3 - 2025-03-02 23:59:59
4 n 2025-03-03 00:00:00

Exit Terminal | $topp | Anzeige | Infos
```

Blockanzeige im Run-Modus

Befehle: **Run** und **Anzeige**

Befehl **#** Infotext anzeigen
Befehl **+cbsl** Blockanfang | Sensor | IR-Leds | Befehlsstring
Befehl **bs** Aufnahme durch Sensor auslösen
Befehl **-** Blockende
Befehl **n** Tagesliste neu berechnen

Bilder mit Bewegungssensor und IR-Leds in der Nacht:

Pin Sensor und IR-Led verwenden

```
// Bildtest3: Fotos, im Zeitraum immer von 7-18 Uhr
{ von="2020-12-01 18:00:01" // Start
  bis="20:59:59" // Ende
  sensor leds // mit IR-Leds
  camctl="-t 5 -n -w 1920 -h 1080" // Befehl
}
{ von="07:00:00" bis="18:00:00" // Start, Ende
  sensor // Sensor ohne IR-Leds
  camctl="-t 5 -n -w 1920 -h 1080" // Befehl
}
{ von="00:00:00" bis="06:59:59" // Start, Ende
  sensor leds // mit Leds
  camctl="-t 5 -n -w 1920 -h 1080" // Befehl
}
```

```
picamctl
Tagesbefehle ab 2025-03-02 10:38:25
Datei: /home/guenther/c/pi/bin/picamctl/bin/_picamctl/cam_test.ctl
Verwendete Pins: 16=Sensor 11=IR-Led

Loop-Befehle:
b Bild aufnehmen
F f Film starten/beenden
s Bewegungssensor ein
l IR-Leds ein
i IR-Filter ein
c Camerabefehl
# Remark
+ Init
- Blockende
n Tagesbefehle neu berechnen
* Rückgabe: Befehl erledigt
! Rückgabe: Fehler

Nr| Bef | Datetime | Info
0 +cbsl 2025-03-02 18:00:01 -t 5 -n -w 1920 -h 1080
1 bs 2025-03-02 18:00:01
2 - 2025-03-02 20:59:59
3 n 2025-03-03 00:00:00
```

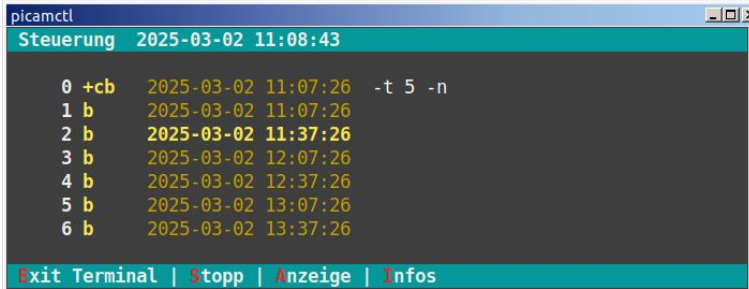
Tagesbefehle anzeigen

Es werden Pin 16 und Pin 11 verwendet

Befehl **#** Infotext anzeigen
Befehl **+cbsl** Blockanfang | Sensor | IR-Leds | Befehlsstring
Befehl **bs** Aufnahme durch Sensor auslösen
Befehl **-** Blockende
Befehl **n** Tagesliste neu berechnen

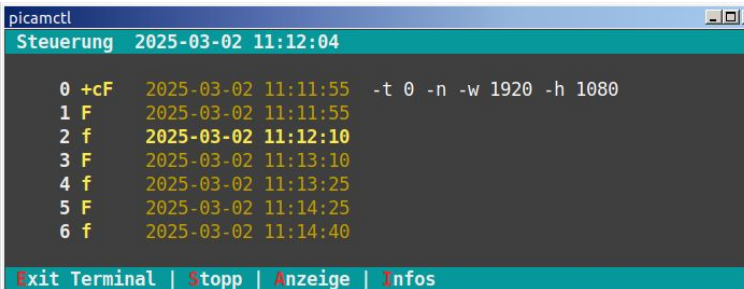
Bilder an bestimmten Wochentagen:

```
// Bildtest4: Fotos, im Zeitraum immer von 7-18 Uhr, alle 30 Minuten
{
  von="07:00:00" // Start
  bis="18:00:00" // Ende
  tag="Di,Fr,So" // Nur Dienstags, Freitags und Sonntags
  wh="30:00" // Wiederholung von Sensor überschrieben
  camctl="-t 0 -n" // Befehl für die Camera
}
```



Aufnahme fortlaufend:

```
// Filmtest1: Filmtest1: Film, endlos, Dauer 15 Sekunden, Pause 1 Stunde
{
  film="15" // Filmdauer 15 Sekunden
  wh="1:0:0" // Wiederholung nach 1 Stunde
  camctl="-t 0 -n -w 1920 -h 1080" // Befehl für die Camera
}
```



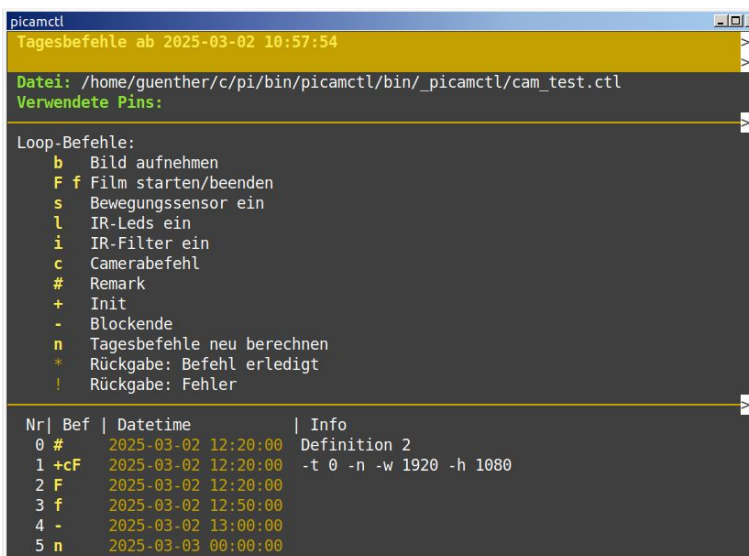
Blockanzeige im Run-Modus

- Befehl +cF Blockanfang | Film | Befehlsstring
- Befehl F Film starten
- Befehl f Film stoppen, nach 15 Sekunden
- Befehl F Film starten, Pause 1 Stunde

Aufnahmen zu bestimmten Zeitpunkten

```
{
  rem="Block 1"
  von="2023-03-31 18:30:00" // fixer Startzeitpunkt
  bis="2024-03-31 19:00:00" // Ende des Zeitraums
  film="30:00" // Filmdauer 30 Minuten
  camctl="-t 0 -n -w 1920 -h 1080"
}
{
  rem="Block 2"
  von="2025-03-02 12:20:00" // fixer Zeitpunkt
  bis="2025-03-02 13:00:00" // Endzeitpunkt
  film="30:00" // Filmdauer 30 Minuten
  camctl="-t 0 -n -w 1920 -h 1080"
}
```

Block 1 ist vom 2023-03-31 bis 2024-03-31 jeweils von 18:30:00 bis 19:00:00 gültig.
 Block 2 ist nur am 2025-03-02 von 12:20:00 bis 13:00:00 gültig.



Keine Pins

- Befehl: +cF Blockanfang | Film | Befehlsstring
- Befehl: F Film starten
- Befehl: f Filmende
- Befehl: - Blockende
- Befehl: n Tagesliste neu berechnen

Fehlersuche

Für die Fehlersuche sollte das Programm ohne Fernbedienung (screen) mit Befehl `picamctl -2` gestartet werden.

Infos | Programm

Zum Überprüfen aller Einstellungen und Verzeichnisse können die Ausgaben vom Befehl `Infos` im Hauptmenu verwendet werden.

Für diese Beispiel wurde in einem PC-Terminal mit `ssh pi@pi8` eine ssh Verbindung zum Raspberry `pi8` und dem Benutzernamen `pi` geöffnet. Mit dem Programmaufruf `picamctl -m` im PC-Terminal wurde die Steuerung am Rechner Pi gestartet.

Befehl: `Infos`

```

picamctl
Programm-Infos
Version      : 0.76      | Programmversion
System      : PC       | Hardware
isXRunning() : true     | Grafische Oberfläche
Pid         : 4933    |
Instanzen   : 1       | Programminstanzen
MALLOC_TRACE : 1     | Heap-Speicher Überwachung 0/1
Config Typ  : a       | Aktuelle Camera-Konfiguration
Simulation  : 1     | Simulationsflag 0/1
WeiterMitTaste: 20   | Maximale Wartezeit nach Fehlern

Logdatei
Obj Log: Daten in Logfile schreiben
LogIsOn : ON
LogOn   : 1 | Var(LogOn)=1 | Logdatei schreiben
Path    : /media/guenther/Scandisk/20250228_160324.log | Logdatei
fLog    : 0x558b9f73f5d0 | Filepointer der Logdatei
LogStart: 2025-02-28 16:03:24 | Startzeit
LogEnd  : - | Endzeit

Fehlerobjekt
Obj Err: 0x558b9f73e690, default
PrintOn : 1 | 1: Print sofort, 0: nur ErrAdd
ScrOn   : 1 | 1: Err Ausgabe auch Bildschirm
LogFile : 1 | 1: Err ins Logfile schreiben
errno   : 0 | Letzter Systemfehler
WarteSec : 20 | Maximale Wartezeit nach Err
Meldungen: 0 | Gespeicherte Err-Meldungen
---

Globale Programm-Variablen
StarOpt   | -2
ProgName  | "picamctl"
WorkDir   | "/home/guenther/c/pi/bin/picamctl/bin"
Config    | "/home/guenther/c/pi/bin/picamctl/bin/_picamctl/picamctl_a.conf"
ConfigDir | "/home/guenther/c/pi/bin/picamctl/bin/_picamctl/"
DatBrowserPi | "mc"
DatBrowserPc | "caja"
TxtEditor  | "nano"
LogOn     | 1
AutoRun   | 1
LogPath   | "/media/guenther/Scandisk/20250228_154538.log"

Globale Config-Variablen 1 | Setup Programm
CamInfoStr | "Pi8, Uhr, Ir-Sensor, Ir-Led, kein Ir-Filter"
CamCtlName | "cam_garten.ctl"
CamOutDirDef | "--tmp/picamctl/"
CamOutDir   | "/home/guenther/tmp/picamctl/"
CamUsbLabel | "Scandisk"
CamLoopShow | 0
CamUhrName  | "PCF8583"
CamUhrSync  | "piclock -s"
CamSensorPin | 16
CamSensorSet | "ir"
CamSensorLab | "Sensor"
CamSensorWait | 2000
CamIrLedPin | 11
CamIrLedSet | "op0"
CamIrLedLab | "IR-Led"
CamIrFlt1Pin | -1
CamIrFlt1Lab | "IR-Flt1"
CamIrFlt2Pin | -1
CamIrFlt2Lab | "IR-Flt2"
CamIrFltSet | "op0"
CamShutterPin | -1
CamShutterSet | "op"
CamShutterLab | "Shutter"
CamSim       | 1
FarbStrCapt | "#

Globale Config-Variablen 2 | Setup Camera A
CamModul    | "imx708"
CamStillName | "camstill.opt"
CamStillOpt  | "-t 0 --width 1920 --height 1080 --saturation 0 --lens-position >
CamStillPre  | "-t 500000 --saturation 0.0 --lens-position 0.5"
CamVidName  | "camvid.opt"
CamVidOpt   | "-t 0 -n -q 100 --width 1920 --height 1080 --saturation 0.0 --le>
CamVidSec   | 5
CamVidPre   | "-t 50000 --saturation 0.0 --height 1080 --width 1920 --lens-pos>

```

Programmversion

Keine grafische Oberfläche

Programmaufrufe werden automatisch auf 1 begrenzt!
Heap-Speicher Überwachung aktiviert

Nur Simulation

Maximale Wartezeit nach Fehleranzeigen

Logdatei

Fehlerobjekt

Fehler sofort ausgeben 0/1

Bildschirmausgabe 0/1. ScrOn=0 geht nur mit Logfile 1
LogFile 1: Fehler ins Logfile umleiten

Auch ohne Fehlerbestätigung weiter nach 20 Sekunden
Derzeit keine gespeicherten Fehlermeldungen

Globale Programmvariablen

Startoption -2 für Debugmodus

Die tatsächlich verwendete Konfiguration
ConfigDir: Ordner für Konfigurationsdateien

Logdatei schreiben

Am Pi die Steuerung automatisch ohne Menu starten
Logdateipfad

Globale Variablen für die Steuerung

CamCtlName: Name der Steuerdatei

CamOutDir: Ordner für Bilder, Filme und Logdatei

CamUsbLabel: Ein verfügbare USB-Speicher wird gemounte und verwendet

CamLoopShow: Anzeigemodus beim Start

CamUhrName: Name der Echtzeituhr

CamUhrSync: Befehl zum synchronisieren der Echtzeituhr

CamSensorPin: Hardwareeinstellungen

CamSensorSet: GPIO-Setup: 'ir' input, Event rising

CamSensorLab: GPIO-Label: 'Sensor'

Weiter Hardware Einstellungen

CamSim: Camerahardware wird simuliert

FarbStrCapt: Farbe für die Dialoge am PC

Die gemerkten Einstellungen für Cameratests

CamModul: Das verwendetes Camera-Modul

CamStillName: Liste der Optionen für Bilder

CamStillOpt: Befehlsstring für libcamera-still

CamStillPre: Optionen für HDMI-Vorschau:

CamVidName: Liste der Optionen für Filme

CamVidOpt: Befehlsstring für libcamera-vid

CamVidSec: Testfilmlänge

CamStillPre: Test-Optionen für libcamera-vid Befehl

Fortsetzung auf der nächsten Seite


```
Globale Config-Variablen 3 | Setup PC Rsync
RemoteHost      | "pi@pi8w"
PiConfigDir     | "/home/pi/.config/picamctl"
RsyncQuelle     | "/media/pi/Scandisk/"
RsyncZiel      | "/home/guenther/tmp/picamctl/"
RsyncWait      | 25
BildBrowser     | "pix -f"
FilmViewer      | "celluloid"
TerminalPc     | "mate-terminal --tab -x"

Verwendete Pins | J8 Header
CamIrlEdPin=11 | IR-Led's Nachtausleuchtung
[11] BCM 17

CamSensorPin=16 | Bewegungssensor
[16] BCM 23

CamShutterPin=-1 | Fernauslöser
▶ nicht verwendet

Screen Check | Status von Programm und Screen-Sitzung
screenstart -i 'picamctl'

Info screenstart 1.28:
Screen      : picamctl | Sitzungsname
Sitzung     : NULL    | Screen PID
ProgPath    : picamctl | Programmpfad ohne Optionen
ProgName    : picamctl | Programmname
ProgOpt     :          | Programm-Optionen
ProgPid     : 4564
DoAttach    : false   | Terminal verwenden
Logdatei    : false   | /tmp/screenstart.log
HostName    : pc780mint
Shell       : /bin/bash
Terminal    : xterm-256color
TTYName     : pts/0
SSH Client  : NULL
Path        : /home/guenther/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr
ExitNr      : 2 | Programm ohne screen
```

Globale PC Einstellungen für Dialog Rsync

RemoteHost: SSH Verbindung zum Pi
 PiConfigDir: Configordner am Pi
 RsyncQuelle: Pi Ordner für Bilder, Filme und Logdateien
 RsyncZiel: PC Zielordner für Rsync
 RsyncWait: Rsync alle 25 Sekunden wiederholen

BildBrowser: Bildbetrachter am PC
 FilmViewer: Videoprogramm am PC
 TerminalPc: Terminalprogramm am PC

Pin Einstellungen

IR-Led's

Bewegungssensor

Programmstatus und Sceen-Sitzung

Siehe: [Screen-Sitzungen](#)

Infos | Laufzeit

Befehl: **Infos** in Loop

```
Loop-Befehle:
b Bild aufnehmen
F f Film starten/beenden
s Bewegungssensor ein
l IR-Leds ein
i IR-Filter ein
c Camerabefehl
# Remark
+ Init
- Blockende
n Tagesbefehle neu berechnen
* Rückgabe: Befehl erledigt
! Rückgabe: Fehler

Loop-Status:
Time : 2025-02-28 22:13:34 | Zeit
Befehl : s* 00:00:00 | Loop-Befehl
SenBef : Fs 00:00:08 | Sensorbefehl
TagBef : - 2025-02-28 23:59:59 | Tagesbefehl
TagNxtI: 9 | Index für nächsten Tagesbefehl
FilmOn : false | Filmaufnahme läuft
WaitSec: 1 | Wartezeit für Taste
LogIsOn: 1 | Logdatei in Verwendung
Debug : 0 | Debugmodus

Exit Terminal|Stopp|Anzeige|Debug|Prog-Check|Log|Test
```

Statusinformationen von Loop

Time	Looptime
Befehl	Aktueller Befehl, * für erledigt
SenBef	Befehl nach Sensor-Event
TagBef	Nächster Tagesbefehl
TagNxtI	Nächster Befehl in der Tagesliste
FilmOn	true, Film läuft
WaitSec	Maximale Wartezeit auf Tasteneingabe
LogIsOn	1, Logdatei schreiben
Debug	0/1 Debugmodus

Screen-Sitzungen

Zur Fernbedienung kann die Steuerung in einer **Screen-Sitzung** mit Befehl **picamctl** gestartet werden. Zur Kontrolle der **Screen-Sitzungen** kann das Hilfsprogramm **screenstart** aus dem Projekt /c verwendet werden. Programm **screenstart** vereinfacht die Schritte um eine passende Arbeitsumgebung zu erzeugen. Siehe [automatischer Programmstart](#).

Mit Befehl **screenstart -i 'picamctl'** kann man alle Infos zum Programm **'picamctl'** in der Screen-Sitzung **'picamctl'** abfragen.

```

pi@pi8:~$ screenstart -i picamctl

Info screenstart 1.28:

Screen      : picamctl      | Sitzungsname
Sitzung     : 4712.picamctl (2025-02-28 11:06:01) (Detached) | Screen PID
ProgPath    : picamctl     | Programmpfad ohne Optionen
ProgName    : picamctl     | Programmname
ProgOpt     :              | Programm-Optionen
ProgPid     : 4713         |
DoAttach    : false       | Terminal verwenden
Logdatei    : false       | /tmp/screenstart.log
HostName    : pi8
Shell       : /bin/bash
Terminal    : xterm-256color
TTYName     : pts/1
SSH Client  : 192.168.178.21 52684 22
Path        : /home/pi/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
sbin:/bin:/usr/local/games:/usr/games
ExitNr      : 1 | Programm und screen Ok

```

Befehlsaufruf

Infos über Screen-Sitzung und Programm

Name der Screen-Sitzung
 Sitzung: Detached für ohne Terminal, Attached mit Terminalfenster
 ProgPath: Programmpfad
 ProgName: Programmname
 ProgOpt: Programmooptionen
 ProgPid: Programm-Id

DoAttach: false, kein Terminalfenster
 Logdatei: true/false. Option für screenstart

HostName: Name des Host Rechners Raspberry Pi

SSH Client: Netz-Verbindung

ExitNr: screenstart -i '...' liefert eine Exitnummer für andere Programme

Exitnummern von **screenstart -i 'Prognose'**

EXIT_Prog1_Screen1	1	PROGRAMM und screen Ok
EXIT_Prog1_Screen0	2	PROGRAMM Ok und kein screen
EXIT_Prog0_Screen0	3	kein PROGRAMM und kein screen
EXIT_NoScreenInstalled	4	screen nicht installiert
EXIT_NoProgInstalled	5	PROGRAMM nicht installiert

Für die genaue Beschreibung siehe Abschnitt [automatischer Programmstart](#).

Automatischer Programmstart

Für die automatische Steuerung der Camera ist es notwendig eine passende Laufzeitumgebung sicherzustellen. Dazu muss eine **Screen-Sitzung** mit genau einer **picamctl** Instanz laufen. Zusätzlich soll im Fehlerfall Programm **picamctl** wieder automatisch neu gestartet werden.

Mit dem Hilfsprogramm **screenstart** kann die Steuerung **picamctl** sicher gestartet und überwacht werden.

Überwachung mit screenstart

Die Steuerung kann mit dem Befehl **screenstart -nC 'picamctl -A -1'** überwacht werden.

screenstart	Das Hilfsprogramm screenstart startet die Steuerung picamctl mit dem Terminalmultiplexer 'screen'. Die Steuerung läuft damit auch ohne Terminal zur Ein- und Ausgabe.
Option -C	Prüfen ob picamctl in einer Screen-Sitzung läuft. Im Fehlerfall wird eine neue Screen-Sitzung mit Befehl ' picamctl -A -1 ' gestartet. Andernfalls wird screenstart mit dem passenden Exit-Code verlassen.
Option -n	Eine neue Screen-Sitzung immer ohne Terminal starten.
picamctl	Das zu startende Programm. Die Klammer '...' ist wegen der Parameter notwendig.
Option -A	Die Steuerung sofort ohne Menus oder Dialoge starten
StartOpt -1	Beim Start mit

Optionen für screenstart

Die Hilfe zum Programm: **screenstart -h**

Mit dem Terminalumschalter 'screen' kann ein Programm im Hintergrund ohne Terminal für die Ein- und Ausgaben gestartet werden.

Das im Hintergrund (detached) laufende Programm kann dann in anderen Terminals zur Bedienung aktiviert (attached) werden.

Mit 'screenstart' kann die Aktivierung und Deaktivierung eines Programms in einer 'screen' Sitzung kontrolliert werden. 'screenstart' erlaubt dabei nur genau eine laufende Programminstanz.

Beispiel: Kontrolle des Programms "picamctl [opt]" mit Aufrufoptionen.

```
screenstart 'picamctl [opt]'
| Aktiviert (attach) das Programm mit screen in einem beliebigen Terminal
| unter dem Sitzungsnamen 'picamctl'. Wenn die Sitzung noch nicht
| existiert, werden alle laufenden Programm-Instanzen gekillt und
| das Programm mit 'picamctl [opt]' in einer screen Sitzung gestartet.

screenstart -n 'picamctl [opt]'
| -n für no Attach
| Deaktiviert (detach) das Programm. Es läuft im Hintergrund weiter.
| Wenn die Sitzung noch nicht existiert, werden alle laufenden
| Programm-Instanzen gekillt und das Programm mit 'picamctl [opt]'
| ohne Terminal in einer deaktivierten screen Sitzung gestartet.

screenstart -i 'picamctl'
| -i für Infos
| Alle Infos zum Programm und zur screen Sitzung.
| 'picamctl' ist der Sitzungsname. Die Programmoptionen [opt]
| sind optional.

screenstart -c 'picamctl'
| -c für nur Check
| Überprüft Programm screen-Sitzung. Der Exitcode kann vom anderen
| Programmen ausgewertet werden. Die Exitcodes findet man weiter unten.
| Exit=1 ist Ok! Eine Programminstanz in screen Sitzung

screenstart -nC 'picamctl [opt]'
| -C für Neustart bei Check (-c) mit Exit!=1
| -n für no Attach
| Dieser Aufruf kann in cron verwendet werden. Damit kann periodisch
| geprüft werden, ob das Programm in der screen-Sitzung läuft.
| Wenn nicht, dann wird das Programm in einer neuen screen-Sitzung
| ohne Terminal gestartet (detached).

screenstart -k 'picamctl'
| -k für kill
| Programm und screen-Sitzung killen
```

Aufrufe:

```
screenstart [Options] [PFAD/]PROGRAMM]
screenstart [Options] '[PFAD/]PROGRAMM [PROG_OPTIONS]'
Weitere Optionen:
-d Debugmodus ein
-h Hilfe 1_read.me anzeigen.
-i Statusinformationen über PROGRAMM und screen
-l Logdatei schreiben.
-k Kill PROGRAMM und Sitzung.
-n No Attach. PROGRAMM detached mit screen starten.
-C Check mit -c. Bei EXIT_Prog1_Screen1 PROGRAMM mit screen
neu starten.
-c PROGRAMM und screen checken. Ergebnis in Exit.
EXIT_Prog1_Screen1 1 PROGRAMM und screen Ok
EXIT_Prog1_Screen0 2 PROGRAMM Ok und kein screen
EXIT_Prog0_Screen0 3 kein PROGRAMM und kein screen
EXIT_NoScreenInstalled 4 screen nicht installiert
EXIT_NoProgInstalled 5 PROGRAMM nicht installiert
```

Linux 'crontab' einstellen

Mit dem Linux-Programm **crontab** können zeitgesteuert Programme aufgerufen werden. Die Einstellungen erfolgen mit Befehl **crontab** im Dialog Setup oder in einem Terminal mit Befehl **crontab -e**

Mit den folgenden crontab-Einstellungen wird sichergestellt, dass die Steuerung nach einem Absturz oder Stromausfall immer wieder gestartet wird.

Einstellungen in Datei crontab:

```
...
SHELL=/bin/sh
PATH=/home/pi/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:

#@reboot screenstart -n 'picamctl -1'
# * Minute | * Stunde | * Tag | * Monat | * Wochentag 0-7 | Befehl
*/1 * * * * screenstart -nC 'picamctl -A -1'
...
```

Erklärungen:

```
| PATH=/home/pi/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
```

Die Pfadangabe **/home/pi/bin** geht davon aus, dass **screenstart** und **picamctl** dort einen Startlink besitzen

```
| PATH      Pfad zu den Programmen
Kommentar # * Minute | * Stunde | * Tag | * Monat | * Wochentag 0-7 | Befehl
Befehl    */1 * * * * screenstart -nC 'picamctl -A -1'
```

/1** : Befehl **screenstart -nC 'picamctl -A -1'** jede Minute wiederholen. Der erste Stern ** steht für Minute. **/** steht für Wiederholen.

Erklärung für **screenstart -nC 'picamctl -A -1'**: Programm **screenstart -C** prüft ob eine **'screen-Sitzung'** mit **picamctl** läuft. Wenn nicht, dann wird eine neue **'screen-Sitzung'** mit **picamctl** gestartet. **-n** für no Attach, also im Hintergrund bleiben. **-A** für AutoRun. **-1** für **'screen-Sitzung'** nur prüfen nicht starten.

Option: Die 'screen'-Sitzung könnte in crontab auch bereits beim Booten mit

```
| @reboot screenstart -n 'gardenctl -1'
```

gestartet werden. Zu diesem Zeitpunkt erfolgt aber das Mounten des USB-Sticks möglicherweise noch nicht zuverlässig.

Logdatei

Logdatei schreiben

Im [Programm | Setup](#) kann die Option Logdatei aktiviert werden. Die Logdatei wird in den Zielordner für Bilder/Filme geschrieben. Wenn die Fehlerumleitung aktiviert wurde, werden Fehler immer in die Logdatei geschrieben. Wenn ein Ein- und Ausgabeterminal vorhanden ist werden die Fehler auch im Terminal angezeigt. Fehlerbestätigungen werden dann immer übersprungen.

Logdatei auswerten

Beispiel: Logdatei vom **20250228_110602.log**

[2025-02-28 11:06:01] picamctl Logstart	Objekt Log wurde erzeugt
Err Test Fehlerumleitung	Fehlerumleitung funktioniert
Steuerdatei: /home/pi/.config/picamctl/cam_garten.ctl	Steuerdatei
[11:06:02] Tagesbefehle compiliert	Tagesbehle aus der Steuerdatei erzeugt
[11:06:02] CamLoopInit	Anzeige, Camera verfügbar, Tagesbefehle, Pins enabled
[11:06:02] Loop-Steuerung läuft	Loop führt die Tagesbefehle aus
[11:06:04] +cbs 2025-02-28 11:06:02	Init Block: + init ,c Camera, b Bild, s Sensor
"-t 1 --width 1920 --height 1080 --saturation 0 --lens-position 0.5 -e jpg"	Befehlsstring Bild aufnehmen
[11:26:40] Exit Programm-Terminal Loop-Steuerung läuft im Hintergrund	Terminal wurde mit Befehl 'Exit Terminal' verlassen
Err Fehler: fd=5 ungültig	
Err GpioEventHandler 9	
Err Bad file descriptor	
Err [2025-02-28 19:00:01] Loop	Fehler in Loop. Pin 9 wurde vermutlich nicht abgemeldet
[19:01:02] +cFsl 2025-02-28 19:01:00	Init Block: + init ,c Camera, F Film, s Sensor, l Leds
"-t 0 -n --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"	Befehlsstring Film aufnehmen
[22:28:27] Exit Programm-Terminal Loop-Steuerung läuft im Hintergrund	Terminal wurde mit Befehl 'Exit Terminal' verlassen
[2025-03-01 00:00:00] Neuer Tag	Neuer Tag
Steuerdatei: /home/pi/.config/picamctl/cam_garten.ctl	Steuerdatei neu einlesen
[00:00:00] Tagesbefehle compiliert	
[00:00:00] CamLoopInit	
[00:00:00] Loop-Steuerung läuft	
[00:00:03] +cFsl 2025-03-01 00:00:01	Init Block +cbs: init, Camera, Bilder, Sensor
"-t 0 -n --width 1920 --height 1080 --saturation 0.0 --lens-position 0.8"	Befehlsstring Film aufnehmen
[07:01:01] +cbs 2025-03-01 07:01:00	Init Block: + init ,c Camera, b Bild, s Sensor
"-t 1 --width 1920 --height 1080 --saturation 0 --lens-position 0.5 -e jpg"	Befehlsstring Bild aufnehmen

Installation

Auf dem Raspberry Pi mit Benutzer `pi` sollte das Projekt `/c` im Ordner `/home/pi/c` entpackt werden.

Die genauen Erklärungen findet man in der Dokumentation:

Projekt `c/` einrichten: www.schmuckhexen.at/programs/c/clar_start.pdf c/1_Dokus/clar_start.pdf

Kurzanleitung

Die ersten Schritte unter: www.schmuckhexen.at/programs/c/clar_start.pdf oder c/1_Dokus/clar_start.pdf

- ▷ 1. Projekt `c/` auf PC oder PI einrichten
- ▷ 2. Projekthilfe `chelp` einrichten
- ▷ 3. Alle Pi Programme compilieren:
 - `cd c/pi` in den Ordner `pi` wechseln
 - `make clean` Programmreste sicher löschen
 - `make` Alle Pi Programme und Hilfsprogramme compilieren
- ▷ 4. Option: Startlink für `picamctl` mit `chelp` einrichten

Startlink | anlegen

Hilfsprogramm `chelp` aufrufen.

Befehl: **Programme/Libraries anlegen/compilieren/ausführen**

Projektname `picamctl` wählen: **Projekt > 8 pi/bin > picamctl/ > RETURN > RETURN**

Programm compilieren: **Compiled**

Startlink anlegen: **Startlink**

Wenn der Startlink eingerichtet wurde, kann das Programm mit dem Befehl `picamctl -2` getestet werden. Ohne Startlink muss der Programmordner beim Start angegeben werden: `c/pi/bin/picamctl/bin/picamctl -2`

Screenstart

Für die Fernsteuerung über eine Internetverbindung mit `ssh` werden noch Hilfsprogramme `screen` und `screenstart` benötigt. Mit dem Befehl `screenstart -i picamctl` kann überprüft werden, ob die Hilfsprogramme einwandfrei funktionieren.

- ▷ `screen` | Terminalumschalter für Linux | Programm installieren mit: `sudo apt-get install screen`
- ▷ `screenstart` | Starter für **Screen-Sitzungen**. Er ermöglicht eine einfache und sichere Bedienung von `screen`.
| Siehe `screenstart -h`. Programmerstellung nachfolgend.

Screenstart | compilieren

Starter für Screen-Sitzungen `screenstart` mit `chelp` compilieren und einrichten. Hilfsprogramm mit Befehl `chelp` aufrufen.

Befehl: **Programme/Libraries anlegen/compilieren/ausführen**

Projektname `screenstart` wählen: **Projekt > 1 bin > screenstart/ > RETURN > RETURN**

Programm compilieren: **Compiled**

Startlink anlegen: **Startlink**

```

pi@pi8: ~
┌───────────┴───────────┐
│ guenther@pc780mint: ~  x  pi@pi8: ~  x  │
└───────────┬───────────┘
chelp: Projektumgebung für C Programm

Dir c/: /home/pi/c | Immer gleich! Realer Ordner oder symb. Link

Projekt : screenstart      | Projektname
Proj Typ : bin             | Allgemeine Programme
Dir      : c/bin/screenstart | Projektordner
Bin      : screenstart     | Programmname
Compiled : true            | Programm compilieren
Run      : c/bin/screenstart/bin/screenstart | Ausführen. Pfad aus makefile
Startlink: /home/pi/bin/screenstart | Startlink anlegen
Make     : c/bin/screenstart/makefile | Anzeigen
DirConf 1: (null)          | Konfigurationen 1.Wahl
DirConf 2: (null)          | Konfigurationen 2.Wahl
Help     : /home/pi/c/bin/screenstart/./1_read.me | Help 1_read.me
Start IDE: ../screenstart.geany | Entwicklungsumgebung laden

| Neues Programm | Edit Libraries | Info |

Befehl | ESC ?

```

Das Programm kann danach in jedem Ordner mit `screenstart -h` aufgerufen werden.

Programmcode

Das Programm wurde in C mit den Hilfsfunktionen aus Projekt [/c](#) und der Entwicklungsumgebung [geany](#) erstellt. Projekt [c/](#) unterstützt die C-Programmierung unter Linux mit über 700 Hilfsfunktionen. Viele der vom Betriebssystem bereitgestellten Funktionen wurden mit einfacheren und robusten Interfaces versehen.

Tasten | Funktionen

Projekt [c/](#) Programme befinden sich zu Laufzeit immer in einer der folgenden nicht blockierenden Keyboardfunktionen.

Die einfachen Keyboardfunktionen liefern Tastencodes vom Typ `uint16_t`. Dabei entsprechen die Werte 0-255 den ASCII-Codes. Die weiteren Codes sind frei definiert für UTF-8 Codes, Funktionstasten, Befehlstasten usw. . Einige Tastencodes sind nur für die interne Kommunikation vorgesehen. Die vollständige Liste der Tastencodes mit Erklärungen findet man in [c/lib/include/iocon.h](#)

Alle Keyboardfunktionen verwenden die private Bibliotheksfunktion `readKbd()` aus [c/lib/iocon/gettaste.c](#). Sie liest die Eingaben aus dem Eingabepuffer des System.

Keyboardfunktionen für Tasten:

```
bool peekTaste(long nsec);           // Verfügbarkeit von Daten auf stdin und Select-Inputs prüfen.
bool peekTasteSec(long sec);        // Funktioniert wie peekTaste(). Timeout in Sekunden.
uint16_t chkTaste(long nsec);       // stdin und Select-Inputs lesen, aber nicht warten. Rückgabe taste_nil bei Timeout
uint16_t chkTasteSec(long sec);     // Funktioniert wie chkTaste(). Timeout in Sekunden.
uint16_t getTaste(const char* Prompt); // Auf Tastencode von stdin warten und Select-Inputs prüfen.
```

Die Keyboardfunktionen verwenden zusätzlich die Linux Funktionen `select/pselect` um die die Ein- oder Ausgabebereitschaft von Gerätedateien, FIFO's und Sockets zu ermitteln. Zum Beispiel liefert der Event-Handler eines Pins am Raspberry nach einem Event die Taste `taste_select`.

Beispiel | Sensorevent

Das folgende Codebeispiel stammt aus der Testfunktion für Bilder mit Bewegungssensor. Siehe: [Menu/Test/Bilder](#). Den gesamten Code findet man im Modul [camlib.c](#) von [picamctl](#).

```
void CamTestBildBSensor(..., int32_t SensorPin, int32_t IrLedPin)
...
CamPinsEnable((SensorPin!=0),(IrLedPin!=0),false,false); // die gewünschten Pins anmelden
CamPinsOpen(NULL); // die Angemeldeten Pins initialisieren

tLoop Loop={
    .Befehl.Ctl=CAMBild, // Steuervariable für Loop
    .Befehl.Time=0, // Befehl: Bild aufnehmen
    .SenBef.Ctl= CAMBild | CAMSensor}; // Sensorbefehl: Bild, Bewegungssensor

if (IrLedPin>0) Loop.Befehl.Ctl |= CAMIrLed; // Option: IR-Led verwenden

GpioEventStart(SensorPin, GPIOEvStartNeu); // Event-Handler für Sensor Pin starten
// In den Keyboardfunktionen Input-Select mit SelectAdd(...) aktivieren

uint16_t Taste=taste_nil; // Taste undefiniert

while(Taste!=taste_esc) // Ende mit ESC-Taste
{
    Taste=low(chkTasteSec(1)); // Warte maximal 1 sec auf eine Taste
    switch(Taste) // Taste auswerten
    {
        case taste_select: // Input-Select: Event-Handler des Bewegungssensors hat ausgelöst
            LoopExecCtl(&Loop); // Befehle ausführen. CAMDone wird gesetzt
            clrCtlDone(&Loop.Befehl); // Clear CAMDone, Befehle weiter verwenden

            GpioEventStart(SensorPin, GPIOEvStartWait); // Debounce-Time abwarten und danach Event-Handler neu starten
            break;

        case 'q': case taste_esc: Taste=taste_esc; // Ende mit ESC oder 'q'
            break;

        default: printf("\n Taste: %s \n",printTaste(Taste)); // Kontrolle: Taste / Funktionstaste anzeigen. Default: taste_nil
    }
}
CamDeletePins();
```

Beispiel | Loop

Der folgende Code zeigt vereinfacht die Hauptschleife der Steuerung:

```

tLoop Loop; // Statusvariable. Speichert alle aktuellen Daten für Loop
            // Loop.Time; LoopTime
            // Loop.Befehl; nächster Loopbefehl: .Time Zeitpunkt, .Str Befehlsstring oder Remark, .Ctl Befehlsflags
            // Loop.SenBef Sensor-Befehl: .Time Zeitpunkt, .Str Befehlsstring oder Remark, .Ctl Befehlsflags
            // Loop.TagBef nächster Tagesbefehl: .Time Zeitpunkt, .Str Befehlsstring oder Remark, .Ctl Befehlsflags
            // Loop.TagNxtI; NIL oder Index für den nächsten TagesBefehl

LoopReload: // Loop neu starten. Z.B. Neustart am Tagesanfang
            // -----

if (CamLoopInit(&Loop) == CAMErr) goto Ende; // LoopRun initialisieren. Im Fehlerfall goto Ende
            // Anzeigemodi prüfen, Camera Typ, verfügbar?, Steuerungs-Datei lesen und
            // die aktuellen Tagesbefehle compilieren
            // Die benötigten Pins werden mit CamPinsEnable() angemeldet.

LogWrite(1,"Loop-Steuerung läuft\n"); // Wenn LogOn, Info loggen

while(true) // Loop: Anzeige, Befehle ausführen, Tagesbefehle und Sensoren auswerten -----
{
    Loop.Time=getLoopSecNow(); // aktuelle Zeit oder Simulationszeit

    // Fehler melden/loggen -----
    ErrChkPrint(Err,tmpStrF("[%s] Loop",getDateStr(Loop.Time)),true); // Loopfehler loggen

    // 1. Nächsten Loop.TagBef holen -----
    if (getTagBef(&Loop)==CAMErr) goto Ende; // nächsten Tagesbefehl lesen
            // Wenn Loop.Befehl und Loop.TagBef erledigt sind,
            // wird der nächsten Tagesbefehls gelesen/ausgewertet.
            // Rückgabe: Loop.Befehl nächster Loopbefehl
            // Loop.SenBef Sensorbefehle
            // Loop.TagBef nächster Tagesbefehl

    // 2. Loop.Befehl prüfen/ausführen -----
    if (ExecBefehl(&Loop)==CAMReload) goto LoopReload; // Loop-Befehle ausführen
            // Rückgabe CAMReload: Neuer Tag, goto LoopReload

    // 3. Loop.TagBef prüfen/ausführen -----
    ChkTagBef(&Loop); // Ende des Tagesbefehls prüfen

    // Anzeige -----
    if (ShowMode==0) CamLoopShowLong(&Loop, getMaxY()); // fortlaufende Anzeige
    else CamLoopShowFix (&Loop, getMaxY()-14); // Blockanzeige
    // Ende Anzeige -----

    // Tasten oder Events auswerten -----
    uint16_t Taste=low( chkTasteSec(WaitSec) ); // maximal WaitSec auf Usereingabe warten

    switch(Taste) // Taste/Event auswerten
    {
        case taste_select: // Sensor-Event -----
            // Sensor-Event wurde ausgelöst
            // Event-Handler wurde angehalten

            if (isSet16(CAMFilm1 | CAMSensor, Loop.SenBef.Ctl))
            {
                // Filmsensor ist aktiv
                // Film läuft bereits
                // Laufzeit verlängern, Filmende neu setzen
                // neues Filmende
                // Sensor-Event nach Ablauf der Debouncezeit starten
                { Loop.Befehl.Ctl = CAMFilm0 | CAMSensor;
                  Loop.Befehl.Time = Loop.SenBef.Time + Loop.Time;
                  GpioEventStart(PinNrSensor ,GPIOEvStartWait);
                }
            }
            else
            {
                // Film neu starten
                // Film mit aktivem Sensor starten
                // Startzeit sofort
                // Loop.FilmOn wird automatisch gesetzt
                { Loop.Befehl.Ctl = CAMFilm1 | CAMSensor;
                  Loop.Befehl.Time = 0;
                }
            }
            // Ende Sensor-Event Film

            if (isSet16(CAMBild | CAMSensor, Loop.SenBef.Ctl))
            {
                // Bildsensor ist aktiv
                // Bild mit Sensorauslösung
                // Startzeit sofort
                Loop.Befehl.Ctl = CAMBild | CAMSensor;
                Loop.Befehl.Time = 0;
            }
        }
        break; // Ende Sensor-Event -----

        case 'e': // Terminal/Programm beenden -----
            Debug=0; // Debugmodus immer beenden
            ExitTerminal(); // Wenn Screen-Sitzung, dann Terminal verlassen
            // Loop fortsetzen

            if (isSet16(CAMBild ,Loop.SenBef.Ctl)) GpioEventStart(PinNrSensor, GPIOEvStartWait);
            if (isSet16(CAMFilm1,Loop.SenBef.Ctl)) GpioEventStart(PinNrSensor ,GPIOEvStartNeu);
            break;

        case 's': LogWrite(1,"Loop-Steuerung Stopp\n"); goto Ende; // Steuerung normal beenden

        case 'a': // Anzeigemodus umschalten -----
            clrScr(); ShowMode++; if (ShowMode>CAMLoopShowMax) ShowMode=0; // Anzeigemodus wechseln
            break;

        case ... // weitere Loop Befehle/Funktionen

    } // Ende switch(Taste)
} // Loop: Ende -----

Debug=0; CamLoopStopp(&Loop);
}

```

Projektdateien

Eine Auflistung der Projektdateien von 'picamctl' findet man in [1_read.me](#):

c/pi/bin	Projekt c/
picamctl/	Projektordner picamctl/
bin/	Ordner Programm
_picamctl/	Ordner Konfiguration
1_read.me	Diese Hilfedatei
2_picamctl.odt	Dokumentation
2_picamctl.pdf	Dokumentation als PDF
picamctl_a.conf	picamctl Konfiguration Cameratyp a
picamctl_b.conf	picamctl Konfiguration Cameratyp b
.	usw.
camstill.opt	Bild-Optionen für libcamera
camvid.opt	Film-Optionen für libcamera
cam_bild_sensor.ctl	*.ctl Steuerungsdateien für die Camera
cam_bild_wh.ctl	
cam_film_sensor.ctl	
cam_garten.ctl	
...	
picamctl	Das fertige Programm
makefile	makefile
picamctl.h	Globale Definitionen
picamctl.c	init(), main(), Menu, Exit()
run.c	Hauptfunktionen aufrufen
runpc.c	Dialoge für PC
cam.h	Dialoge und Einstellungen für Camera
cam.c	
camlog.h	Logdatei schreiben
camlog.c	
camlib.h	Steuerbefehle für das Programm 'libcamera'
camlib.c	
camctl.h	Camera Steuerung. Steuerdatei lesen/parsen
camctl.c	und Liste der Tagesbefehle erzeugen
camloop.h	Loop-Steuerung: Tagesbefehle ausführen
camloop.c	
@gpioci.h	Linkbibliothek c/pi/bininc
@gpioci.c	Lib-Link: I/O-Pins mit Chip-Interface /dev/gpiochip0 steuern
@gpioci_pi.h	Lib-Link: Source
optdlg.h	Lib-Link: Kopie von <linux/gpio.h> vom Pi für die Simulation am PC
optdlg.c	Option-Dialog. Verwaltung von Programmoptionen für 'libcamera'
...	
picamctl.geany	Starter für IDE geany

Raspberry Pi Boot-Einstellungen

Einstellungen in /boot/config.txt

Camera Konfiguration in /boot/config.txt

You may need to alter the camera configuration in your /boot/config.txt if:

- You are using a 3rd party camera (the manufacturer's instructions should explain the changes you need to make).
- You are using an official Raspberry Pi camera but wish to use a non-standard driver/overlay.

If you do need to add your own dtoverlay, the following are currently recognised.

Camera Module	In /boot/config.txt
V1 camera (OV5647)	dtoverlay=ov5647
V2 camera (IMX219)	dtoverlay=imx219
HQ camera (IMX477)	dtoverlay=imx477
GS camera (IMX296)	dtoverlay=imx296
Camera Module 3 (IMX708)	dtoverlay=imx708
IMX290 and IMX327	dtoverlay=imx290,clock-frequency=74250000 or dtoverlay=imx290,clock-frequency=37125000 (both modules share the imx290 kernel driver; please refer to instructions from the module vendor for the correct frequency)
IMX378	dtoverlay=imx378
OV9281	dtoverlay=ov9281

To override the automatic camera detection, Bullseye users will also need to delete the entry camera_auto_detect=1 if present in the config.txt file. Your Raspberry Pi will need to be rebooted after editing this file.

Zusammenbau der Hardware

Netzteile einbauen

Basisrohre mit schwarzer Halterung für Komponenten.



Option: Mobile Stromversorgung mit 2 Akkus vom Typ Einhell PowerX 3.0Ah, 18V.



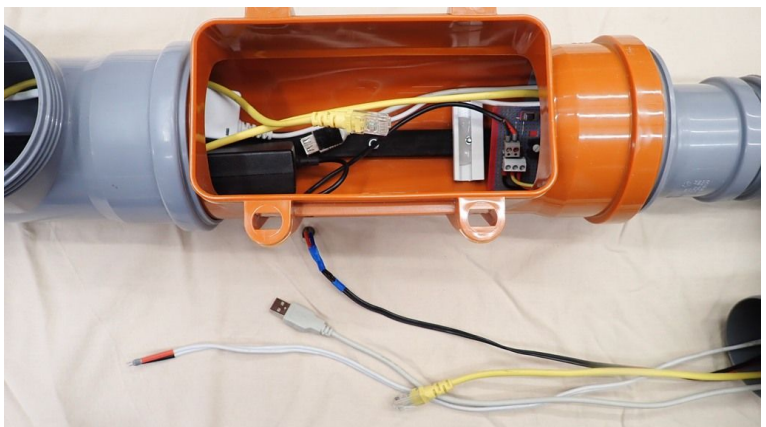
Option: Zwei Schaltnetzteile

Stromversorgung 5V für Pi: Kabel schwarz
Stromversorgung 12V, Leds: Kabel weiß



Alle Kabel:

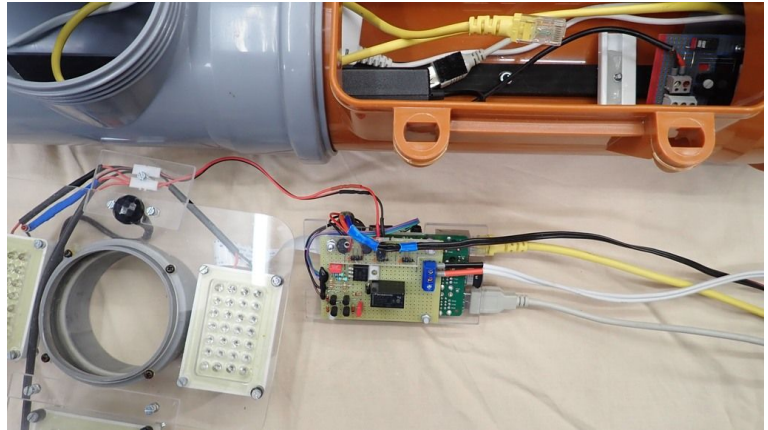
Stromversorgung 5V für Pi: Kabel schwarz
Stromversorgung 12V, Leds: Kabel weiß
Netz kabel: Kabel gelb
USB-Verlängerung: Kabel grau



Steuerblock anschließen

Alle Verbindungen: Ansicht von unten.

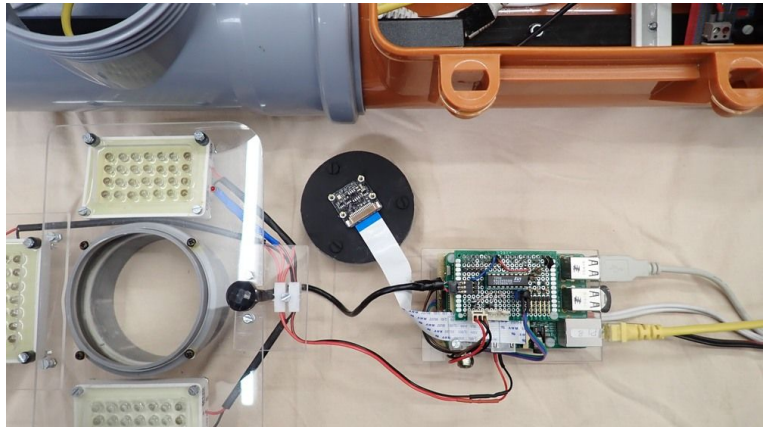
Led-Halterung mit Bewegungssensor.
Steuerung von unten: 12V Relais usw.



Ansicht von oben ohne Echtzeituhr:

Verteilerplatine `PI_CAMERA_J8` für Pi-Header J8
mit Treiber `ULN_2803` (8 Open Collector)
Sensoranschluss schwarz, verpolungssicher.

Camera und Kabelsicherung am Pi.

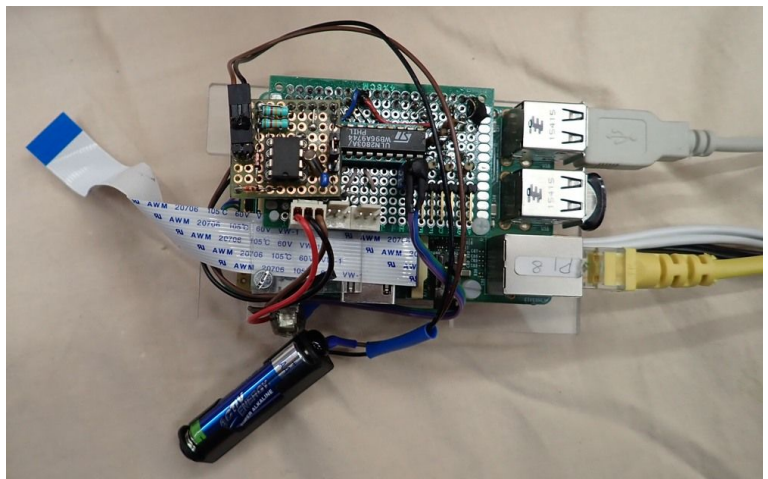


Rohrende mit Halterung für den Steuerblock.



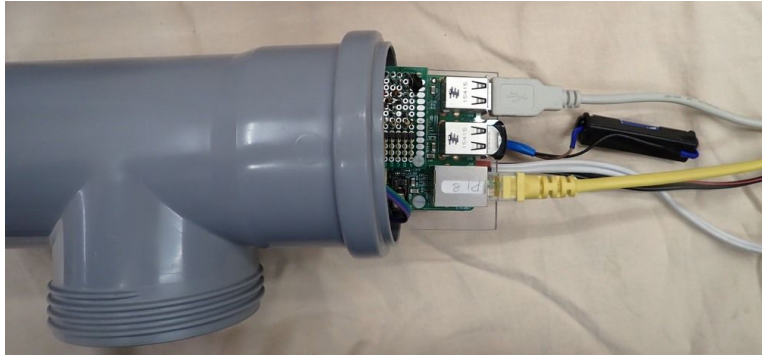
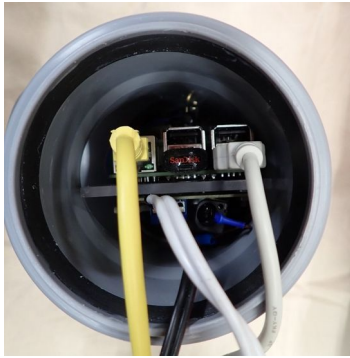
Steuerblock:

Ansicht von oben mit Echtzeituhr und
1.5V Stützbatterie AAA.
Anschluß verpolungssicher.



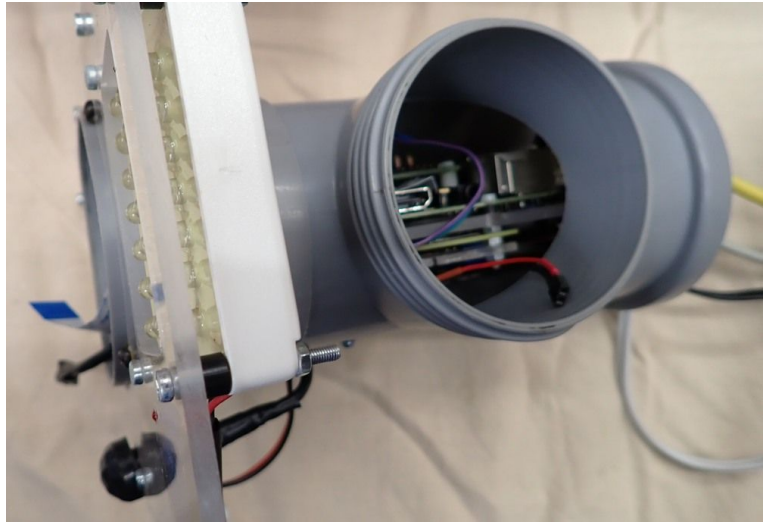
Endmontage

Einschub ins Rohrende



Ledhalter mit Sensor aufgeschoben.

12V Leitung für Leds eingefädelt:
Stecker rot/schwarz,

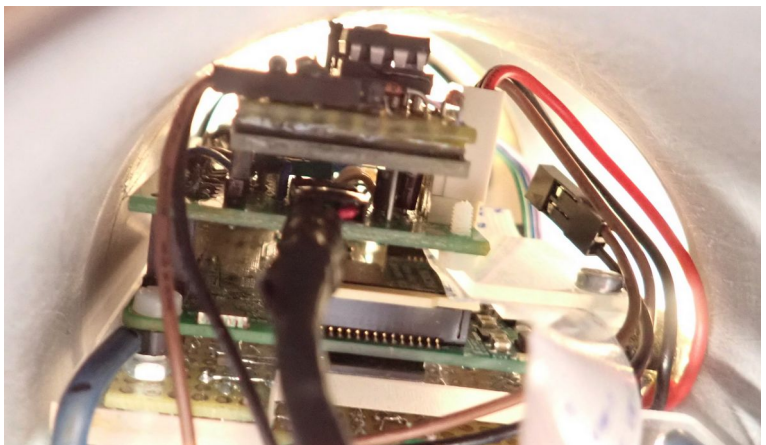


12V Leitung für Leds angesteckt:
Markierung rot/schwarz für Polung beachten.

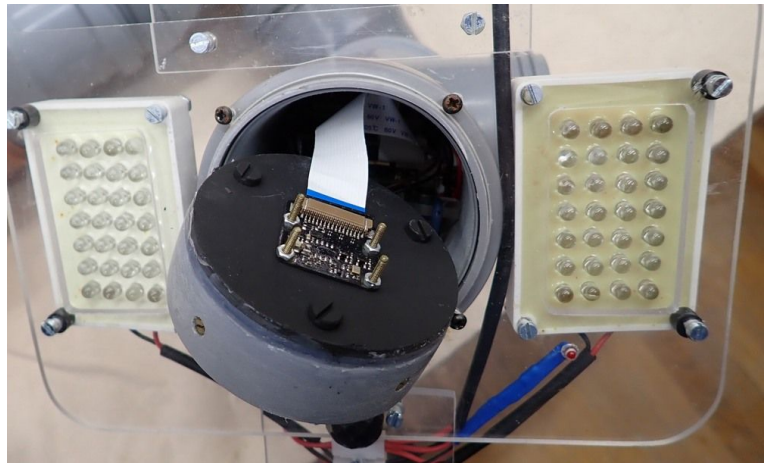


Den dreipoligen Stecker (weiblich) der schwarzen
Sensorleitung auf Platine [PI_CAMERA_J8](#) anstecken.

Anschluss verpolungssicher.



Camera-Modul mit Steuerblock verbinden



Camera-Modul von vorne aufstecken.
IR-Leds für das Weitwinkelobjektiv einstellen.



Fertig



Dokus zum Projekt /c

Die allgemeinen Dokumentationen findet man unter [c/1_Dokus](#) oder im Internet:

Vorwort:	www.schmuckhexen.at/programs/c/clar_vorwort.pdf	c/1_Dokus/clar_vorwort.pdf
Projekt c/ einrichten. Die ersten Schritte:	www.schmuckhexen.at/programs/c/clar_start.pdf	c/1_Dokus/clar_start.pdf
Projekthilfe und Projektmanager:	www.schmuckhexen.at/programs/c/clar_chelp.pdf	c/1_Dokus/clar_chelp.pdf
Ein neues C Programm erstellen:	www.schmuckhexen.at/programs/c/clar_projekt.pdf	c/1_Dokus/clar_projekt.pdf
Basisobjekte ohne Terminal In/Ouput:	www.schmuckhexen.at/programs/c/clar_objekte1.pdf	c/1_Dokus/clar_objekte1.pdf
Terminalsteuerung Box-Objekte für In/Ouput:	www.schmuckhexen.at/programs/c/clar_objekte2.pdf	c/1_Dokus/clar_objekte2.pdf

GNU General Public License

```
/*
 * Copyright 2023-2025 Günther Schardinger <v.schardinger@gmx.net>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
 * MA 02110-1301, USA.
 */
```