RHEINISCHE FACHHOCHSCHULE KÖLN

University of Applied Sciences

Fachbereich: Elektrotechnik

Studiengang: Bachelor of Engineering

Projektarbeit PA 7

Entwicklung eines Kamerasystems mit EtherCat Schnittstelle

Projektarbeit vorgelegt von: Leo Enns
Mat.-Nr. BE02132052

Prüfer: Dr., Dipl. -Ing(FH) Dusko Lukac MBA, M.Eng.

Sommersemester 2018

Inhalt

Abkürzungsverzeichnis

Abkürzung	Darstellung des abgekürzten Begriffes
AWL	Anweisungsliste
CPU	Central Processing Unit
CSI	Camera Serial Interface
DSI	Display Serial Interface
EtherCat	Ethernet for Control Automation Technology
FBS	Funktionsbausteinsprache
FPGA	Field Programmable Gate Array
FPS	Frames per second
Gb	Giga bit
GB	Giga byte
Gbps	Giga bit per secound
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDMI	High Definition Multimedia Interface
I²C	Inter-Integrated Circuit
KOP	Kontaktplan
Mb	Mega bit
MB	Mega byte
MCU	Microcontroller Unit
MPU	Microprocessor Unit
OS	Operating System
px	Pixel
RJ45	Registered Jack 45
SBC	Single Board Computer
SPI	Serial Peripheral Interface
SSH	Secure Shell
ST	Strukturierter Text
USB	Universal Serial Bus
WLAN	Wireless Local Area Network

1 Einleitung

In dieser Projektarbeit wird die Implementierung eines DIY (Do it yourself) Kamerasystems beschrieben.

Ausgangspunkt für diese Projektarbeit ist meine Bachelorthesis „***Untersuchung der EtherCat-Schnittstelle für Industrieroboter und Realisierung einer Kameraanbindung“***. Für die besagte Bachelorthesis ist ein passendes Kamerasystem notwendig.

Mit diesem Kamerasystem soll es möglich sein, Bilder zu erstellen und diese auszuwerten. Außerdem muss es eine Möglichkeit geben, Daten mit einer Kuka KRC4 Steuerung auszutauschen. Dafür soll der EtherCat Bus verwendet werden. Die Implementierung des Bussystems wird jedoch erst in der Bachelorthesis behandelt. Wodurch nur eine passende Hardware verbaut werden muss.

Das Finale System soll mittels der Bildauswertung, Objekte erkennen und deren Position in Koordinaten Form an den Roboter schicken. Der Roboter soll anhand der Koordinaten das Objekt anfahren und mit einem Greifer aufnehmen können.

Das Kamera System muss auf Open Source Software basieren damit, der Quellecode für Schulungszwecke einsehbar ist und in zukünftigen Projektarbeiten modifiziert werden kann.

Dieses System muss dafür folgende Bedingungen erfüllen:

- Kostengünstig Komponenten
- Einen Controller, der variable programmiert und genutzt werden kann
- Passendes Objektiv und Kameramodul um diverse Objekte zur erkennen
- Gegeben falls eine passende Beleuchtung um die Objekte besser zu erkennen
- EtherCat Schnittstelle (Slave)
- Stabile Kamerahalterung zur Befestigung am Kuka Testwagen des Robotik Labor der RFH

2 Vorbereitung\Grundlagen

2.1 Auswahl der Hardware

2.1.1 Controller

Das Herz des Kamerasystems ist der Controller. Um diesen zu realisieren gibt es verschieden Möglichkeiten:

2.1.1.1 FPGA (Field Programmable Gate Array)[1]

Mit einem FPGA lassen sich logische Schaltungen variabel über eine Programmiersprache erstellen. Der FPGA arbeitet somit nicht zyklisch ein Programm ab, sondern verknüpft direkt und ohne Systemtakt die Eingänge über die logische Schaltung mit den Ausgängen. Auch bei der Bildverarbeitung können FPGAs verwendet werden um den Bit stream eines Bildes direkt zu verarbeiten und weiter zu geben. Jedoch werden ggf. mehrere FPGAs benötigt um verschieden Prozesse zu realisieren. Und das Angebot an Schnittstellen für den FPGA sind nicht umfangreich. Einen zyklischen Programmablauf oder sogar ein Betriebssystem sind auf einem FPGA nicht realisierbar.

2.1.1.2 Mikrocontroller (MCU)

Der Mikrocontroller hat eine feste Logik Struktur und bieten einen Befehlssatz an, mit dem ein zyklisches Programm erstellt werden kann. MCUs finden oft ihre Anwendung auf dedizierten Komponenten eines größeren Systems. Beispielsweise als Buscontroller für einen USB Port oder EtherCat Slave Port, aber auch als kompakte Auswerteeinheit auf einem Sensor.

Es gibt aber auch Anwendungen in denen ein MCU eigenständig läuft. Wie zum Beispiel in einem Arduino Board. Hier kommen häufig MCUs vom Typ ATmega2560 zum Einsatz. Der ATmega2560 verarbeitet Befehlssätze mit 8 Bit mit einer Frequenz von 16 MHz. Das ist zwar im Verhältnis zu modernen Prozessoren mit 64 Bit und 3 GHz deutlich schwächer, jedoch sind diese System schlank und haben keine zusätzlichen Prozesse (wie ein Betriebssystem usw.) wodurch sie viel schneller starten und auch schnell schalten können.

2.1.1.3 Mikroprozessor (MPU)

Der Mikroprozessor ist für die Verarbeitung von großen Datenmengen mit hohen Geschwindigkeiten gedacht. Auch hier ist die Logik Struktur (Prozessorarchitektur) fest und es gibt feste Befehlssätze zur Steuerung und Programmerstellung. Jedoch sind ist die Datenbreite mit 32 oder 64 Bit sowie auch die Taktfrequenz mit 3 GHz deutlich höher wie beim MCU. Mikroprozessoren werden meistens eingesetzt um mehrere Komponenten zu verbinden. Man spricht dann auch von der CPU (Central Processing Unit). Beim Einsatz von einem Betriebssystem (Operating System „OS“) ist eine CPU notwendig sowie auch eine GPU (Graphics Processing Unit) damit auch eine Benutzeroberfläche (Graphical User Interface „GUI“) für den Benutzer angezeigt werden kann.

Zwei bekannte Prozessorgruppen sind X86 und ARM. Die Unterschiede bestehen in der Prozessorarchitektur und tendenziell wird X86 bei Laptop und PCs eingesetzt. Bei mobilen Geräten und Mini-PCs kommt vermehrt der Einsatz von ARM vor.

[1] Vgl. Prof. Dr.-Ing. habil. Karl-Heinz Meusel, „Digitaltechnik - FPGA“, WS 2016.

Abhängig vom verwendeten OS sind auch nicht alle MPUs verwendbar. Ein Windows OS benötigt einen Prozessor der X86 Familie. Bei einem Linux OS ist es abhängig vom eingesetzten Kernel welche Prozessoren verwendet werden können.

2.1.1.4 Raspberry Pi

Das Steuergerät muss in der Lage sein ein Kameramodul zu breitreiben, somit ist eine passende Schnittstelle für eine Kamera nötig. Desweitern müssen die Bilder innerhalb des Controllers ausgewertet werden und das setzt Voraus das der Controller ein OS mit GUI betrieben kann. Somit sind MCU und FPGA nicht geeignet für den Anwendungsfall. Die Endscheidung, ob X86 oder ARM ist oft eine Preisfrage. Mini – PCs mit ARM Prozessoren kosten oft nur ein Bruchteil von einem vollwertigen PC auf X86 Basis.

Der Raspberry PI Model 3 B ist ein SBC (Single Board Computer) oder Einplatinencomputer von der britischen Raspberry PI Fundation. Er wurde entwickelt als Entwicklungsumgebung für den Einstieg in die Programmierung. Dieser Einplatinencomputer bietet alle Schnittstellen, die wir benötigen.

Das Betriebssystem für Raspberry Pi ist Raspbian, eine Linux Distribution und ist somit auch Open Source basiert wodurch keine zusätzlichen Kosten anfallen. In Linux können auch Programme mit verschieden Programmiersprachen, wie C ,C++ und Python, erstellt werden.

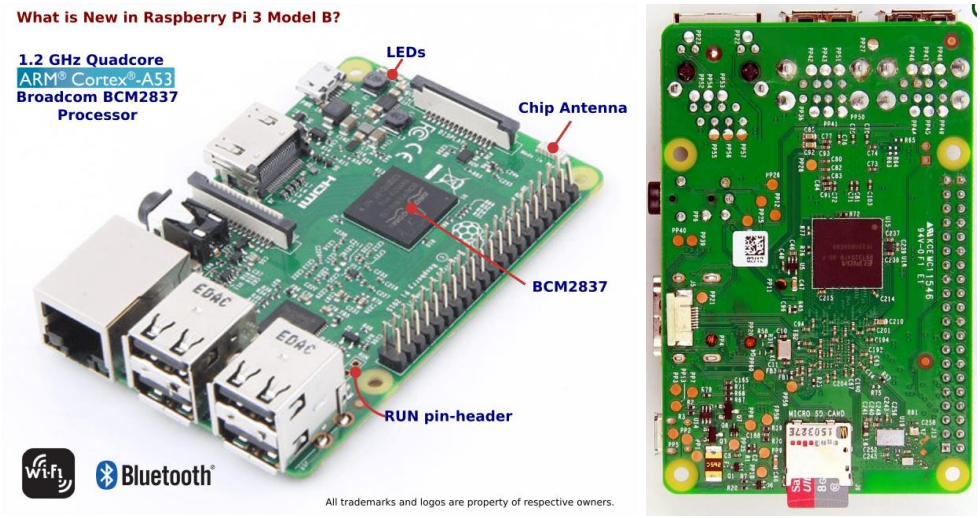


Abbildung 2-1 - Raspberry Pi Model 3 [2]

[2] wiki.seeedstudio.com, „Raspberry Pi 3 Model B“, zugegriffen 6. Juni 2018, http://wiki.seeedstudio.com/Raspberry_Pi_3_Model_B/.

4

Technische Daten: [3]

- CPU: Quad-Core-Prozessor mit 1,2 GHz
- GPU: Dual-Core-GPU VideoCore IV mit OpenGL ES 2.0 und OpenVG mit Hardwarebeschleunigung und 1080p30 H.264 High-Profile-Decoding
- Arbeitsspeicher: 1 GByte LPDDR2-SDRAM
- WLAN: BCM43143 onboard für IEEE 802.11b, g und n im 2,4 GHz-Bereich
- Bluetooth: Bluetooth Classic und Low Energy (BLE) onboard (Bluetooth 4.1
- RJ45-Ethernet-Anschluss mit 10/100 MBit/s
- 4 x USB-2.0-Anschlüsse
- 4-poliger Klinkenstecker mit Stereo-Ausgang und Composite-Video
- HDMI-Anschluss
- 15-poliger MPI-CSI-2-Steckverbinder für HD-Videokamera
- 15-poliger serieller Display-Schnittstellensteckverbinder (DSI), z.B. für das offizielle Touchscreen-Display für den Raspberry Pi
- 40-polige GPIO-Stiftleiste mit seriellen Bussen

Der Raspberry Pi kommt ohne Gehäuse, somit ist es notwendig hierfür etwas Passendes zu finden. Auch um die Kühlung muss man sich kümmern. Bei der Verarbeitung von Bildmaterial werden GPU und CPU schnell zu warm, somit ist eine gute Kühlung wichtig. In diesem Fall wird ein 5 V Lüfter mit 0,16A an der GPIO Schnittstelle des Raspberry PIs verbunden und zusätzlich noch Passive Kühlkörperrippen auf die Chips geklebt.

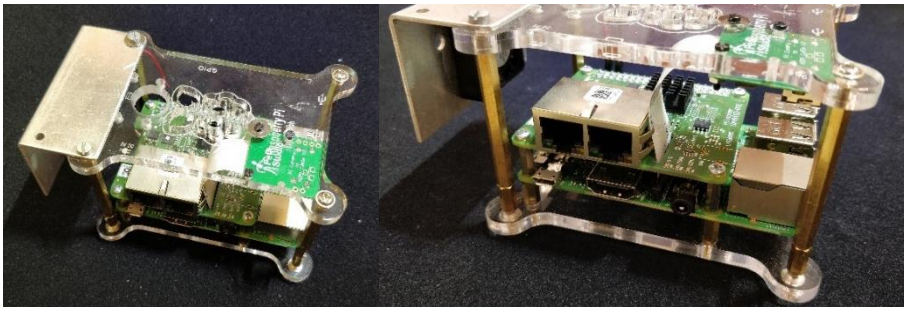


Abbildung 2-2 - Raspberry Pi Gehäuse[4]

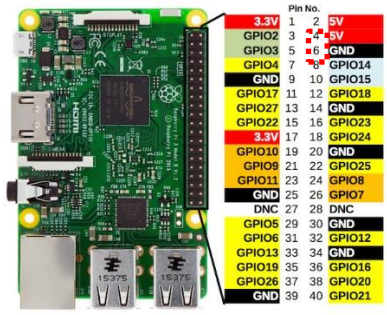


Abbildung 2-3 - Raspberry PI GPIO [5]

---

[3] Patrick Schnabel, „Raspberry Pi 3 Modell B“, zugegriffen 6. Juni 2018, https://www.elektronik-kompendium.de/sites/raspberry-pi/2102291.htm.

[4] Quelle: Eigene Aufnahmen

[5] Steve Chamberlin, „Raspberry Pi GPIO Programming in C | Big Mess o’ Wires“, zugegriffen 7. Juni 2018, https://www.bigmessowires.com/2018/05/26/raspberry-pi-gpio-programming-in-c/.

2.1.2 Kamera

Nach der Auswahl des Controllers muss auch eine passende Kamera gefunden werden. Kriterien für die Auswahl sind die Schnittstelle, Auflösung und das Objektiv.

2.1.2.1 Schnittstelle

Die Schnittstelle ist vom Raspberry PI vorgegeben mit CSI vorgeben. Es wäre aber auch möglich eine USB Kamera am Raspberry PI anzuschließen, jedoch haben die meisten USB Kameras gerade mal eine Auflösung von 2 Megapixel. Desweitern muss auch noch erwähnt werden, dass beim Anschluss einer USB Kamera am Raspberry Pi, die CPU Last erhöht wird. [6]

Die CSI-2 (Camera Serial Interface) Schnittstelle ist eine schnelle Schnittstelle auf dem Raspberry Pi mit 15 Kontakten. Über diese Schnittstelle ist eine theoretische Brandbreite von 2,5 bis 2,9 Gbps möglich[7].

Ein Bild mit der Auflösung von 1920 x 1080 und 3 Farbtiefen mit je 8 Bit (RGB von je 0 – 256) erzeugt somit schon eine Bildgröße von $1920\,px * 1080\,px * 24\,Bit = 49{,}77\,\text{Mbit}$. Bei einem Video mit 30 Fps wird also schon eine Bandbreite von 1,5 Gbps benötigt.

$$1920\,px * 1080\,px * 24\,Bit * 30\,Fps = 1500\frac{\text{Mbit}}{s} = \frac{1500}{8} = 187{,}5\frac{MByte}{s}$$

Eine USB 2.0 Verbindung hat eine theoretische Bandbreite von 480 Mbps und könnte das Videosignal nicht ohne vorherige Komprimierung übertragen. Dies wäre ein weiter Grund gegen eine USB Kamera.

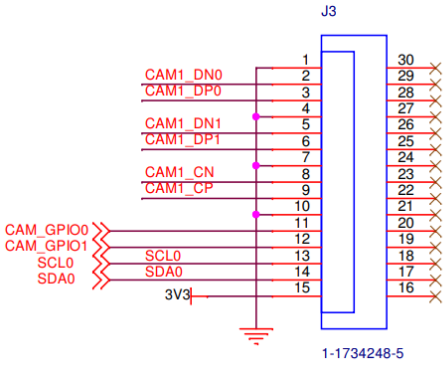


Abbildung 2-4 - Raspberry CSI[8]

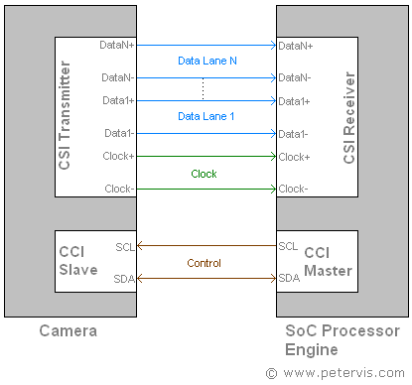


Abbildung 2-5 - CSI Schema[9]

---

[6] Vgl. Harald Schmidt, *Raspberry Pi programmieren mit C/C++ und Bash: Mehr als 50 Programme rund um Foto, Video & Audio* (München: Hanser, 2018), 518.

[7] Vgl. Alexander Stohr, „Camera Serial Interface“, *Wikipedia*, 7. Februar 2018, https://en.wikipedia.org/w/index.php?title=Camera_Serial_Interface&oldid=824381676.

[8] Raspberry Pi Foundation, „Schematics Raspberry Pi“, zugegriffen 7. Juni 2018, https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3b_1p2_reduced.pdf.

[9] Peter J. Vis, „Raspberry Pi CSI-2 Connector Specifications“, zugegriffen 7. Juni 2018, https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI-2_Connector_Specifications.html.

2.1.2.2 Kameramodule und Objektive

Von der Raspberry PI Fundation gibt es auch passende Kameras. Das aktuellste Model V2 bietet folgende technische Daten: [10]

- Standbild-Auflösung 8 Megapixel
- Video-Modi 1080p30, 720p60 und 640 x 480p60/90
- Sensor Sony IMX219
- Sensor-Auflösung 3280 x 2464 Pixel
- Sensor-Fläche 3,68 x 2,76 mm (Diagonale 4,6 mm)
- Pixel-Größe 1,12 x 1,12 µm
- Optische Größe 1/4"
- Brennweite **3,04 mm**
- Blickwinkel (horizontal) 62,2°
- Blende F 2.0
- Gewicht 3 g
- Abmessungen 25 x 24 x 9 mm

Der Vorteil von diesem Modul ist, dass hochauflösende Bild möglich sind und das bei einer sehr kompakten Bauform. Der Nachteil hier ist das man das Objektiv nicht austauschen kann und der Fokus fix ist. Das Modul wird zudem in einer Variante mit und ohne Infrarot Filter angeboten. Ohne Infrarotfilter könnten somit auch Nachtaufnahmen gemacht werden.

Abbildung 2-6 – Raspberry Pi Camera V2[11]

Eine Alternative dazu ist das Kameramodul Arducam Rev.C OV5647 CS Mount. [12]

- Standbild-Auflösung 5 Megapixel
- Video-Modi 1080p30, 720p60 und 640 x 480p60/90
- Sensor OmniVision OV5647
- Sensor-Auflösung 2592 × 1944 pixels
- Sensor-Fläche 3.67 × 2.74 mm
- Pixel-Größe 1.4 µm × 1.4 µm
- Optische Größe 1/4

---

[10] Andreas Potthoff, „Die Raspberry Pi Kamera Module – Inbetriebnahme und Verwendung | ElectroDrome“, zugegriffen 7. Juni 2018, https://electrodrome.net/die-raspberry-pi-kamera-module-inbetriebnahme-und-verwendung/.

[11] Quelle: Eigene Aufnahme

[12] Andreas Potthoff, „Die Raspberry Pi Kamera Module – Inbetriebnahme und Verwendung | ElectroDrome“.

Auf diesen Sensor können verschiede Objektive installiert werden: [13]

LS-2716 4mm

- Brennweite 4 mm
- Blickwinkel (horizontal) 81°
- Blende F 1,4

Abbildung 2-7 - Arducam mit 4mm Objektiv [14]

0612-3MP-A 6mm

- Brennweite 6 mm
- Blickwinkel (horizontal) 72°
- Blende F 1,2

Abbildung 2-8 - Arducam mit 6mm Objektiv[15]

Der Vorteil von diesen Modulen ist das die Objektive wechselbar sind und der Fokus am Objektiv Einstellbar ist. Jedoch ist die Auflösung etwas geringer. Weil das zu planenden Kamerasystem auch als Schulungsstation genutzt werden soll sind alle Objektive interessant um verschieden Anwendungsfälle zu realisieren.

[13] Ebay, „2x Arducam OV5647 CS Mount Kameramodul für Raspberry Pi mit 4 und 6mm Objektiv“, eBay, zugegriffen 7. Juni 2018, https://www.ebay.de/itm/2x-Arducam-OV5647-CS-Mount-Kameramodul-fuer-Raspberry-Pi-mit-4-und-6mm-Objektiv-/263647321059.

[14] Quelle: Eigene Aufnahme

[15] Quelle: Eigene Aufnahme

2.1.2.3 Objektiv

Die Brennweite der Objektive ist später entscheidet wie hoch die Kamera über den zu erkennenden Objekten zu installieren ist. Dieser Abstand kann wie folgt berechnet werden:

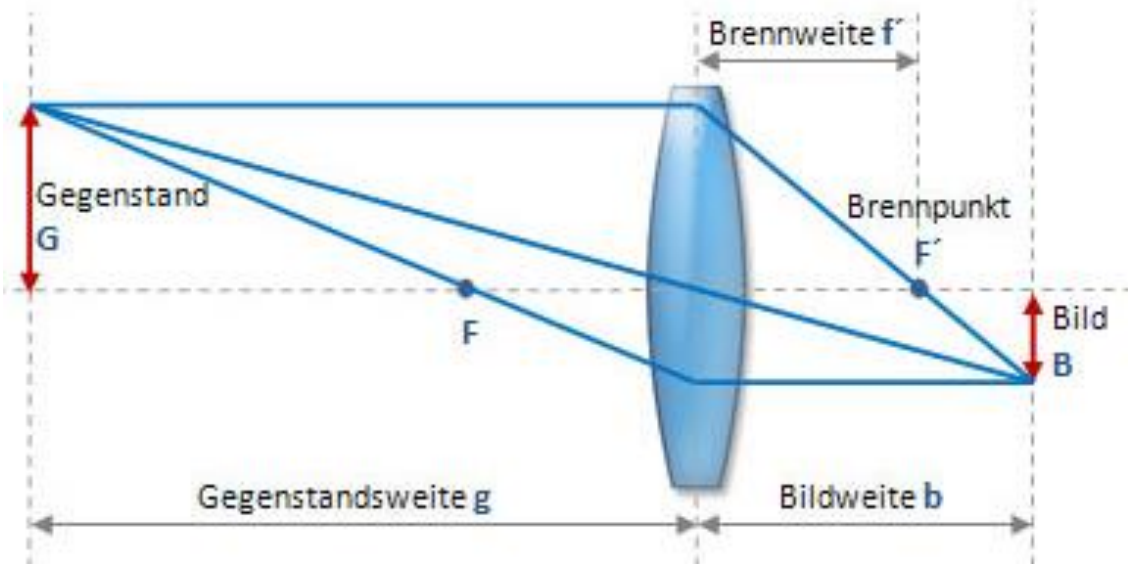


Abbildung 2-9 - Konvexe Linse[16]

Zur Berechnung des benötigten Abstands kann die folgen Formel verwendet werden

$$g = f' * \left(\frac{G}{B} + 1\right)$$ [17]

Formel 2-1 - Objektivabstand

Mit der Raspberry Pi V2 Kamera ergibt sich bei einem gewünschten Sichtfeld von 620 mm Breite folgender Abstand:

$$g = 3{,}04\ \text{mm} * \left(\frac{620\ mm}{3{,}68\ \text{mm}} + 1\right) = 515\ mm$$

Die Höhe des Sichtfeldes lässt sich durch folgende Formel berechnen:

$$G = B * \left(\frac{g}{f'} - 1\right)$$ [18]

Formel 2-2 - Bildgröße

$$G = 2{,}76\ \text{mm} * \left(\frac{515\text{mm}}{3{,}04\ mm} - 1\right) = 465\ \text{mm}$$

[16] Lars Fermum, „Optische Grundlagen“, zugegriffen 7. Juni 2018, http://www.vision-doctor.com/optische-grundlagen.html.

[17] Vgl. Lars Fermum.

[18] Vgl. Lars Fermum.

Daraus entsteht ein mögliches Sichtfeld von 620 x 465 mm im Verhältnis 4:3 bei einem Abstand von 515 mm

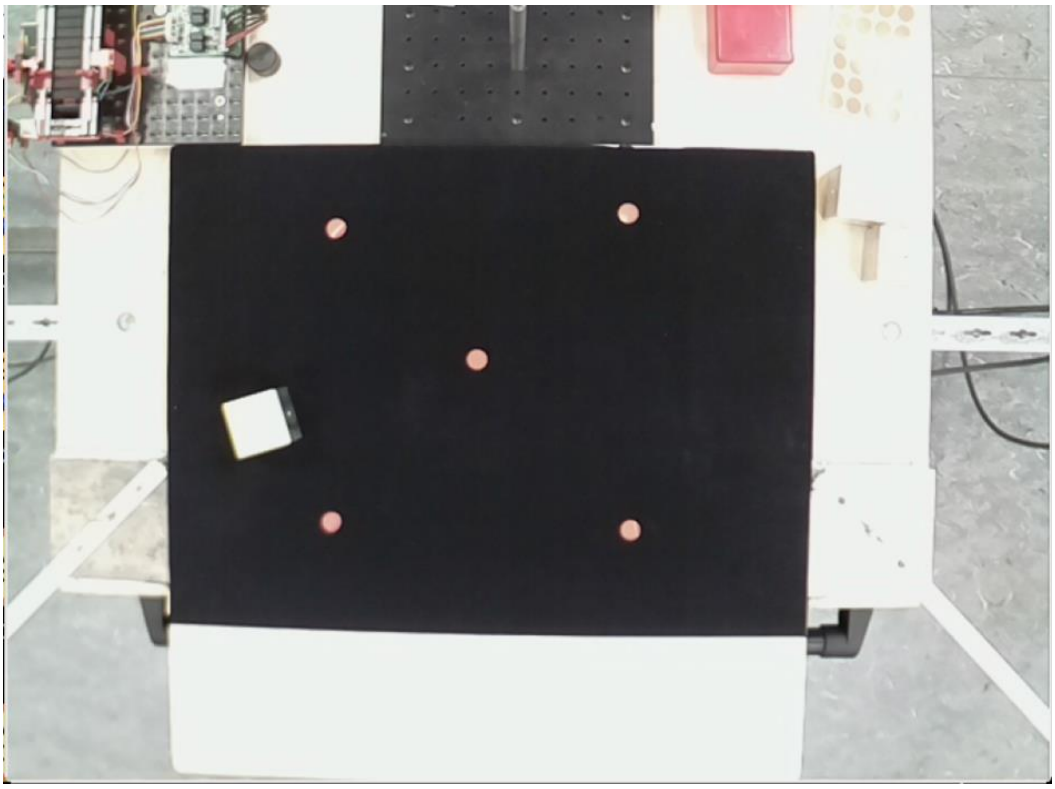


Abbildung 2-10 - Raspberry Pi Camera V2 aus 1000 mm Höhe[19]

Für die Arducam mit dem 4 mm Objektiv

$$\mathrm{g} = 4\ \mathrm{mm} * \left(\frac{620\ mm}{3{,}67\ \mathrm{mm}} + 1\right) = 679\ mm$$

$$G = 2{,}74\ mm * \left(\frac{679\ \mathrm{mm}}{4\ mm} - 1\right) = 462\ mm$$

Daraus entsteht ein mögliches Sichtfeld von 620 x 462 mm im Verhältnis von ca. 4:3 bei einem Abstand von 679 mm

Für die Arducam mit dem 6 mm Objektiv

$$\mathrm{g} = 6\ \mathrm{mm} * \left(\frac{620\ mm}{3{,}67\ \mathrm{mm}} + 1\right) = 1019\ mm$$

$$G = 2{,}74\ mm * \left(\frac{1019\ \mathrm{mm}}{6\ mm} - 1\right) = 462\ mm$$

Daraus entsteht ein mögliches Sichtfeld von 620 x 462 mm im Verhältnis von ca. 4:3 bei einem Abstand von 1019 mm

[19] Quelle: Eigene Aufnahme

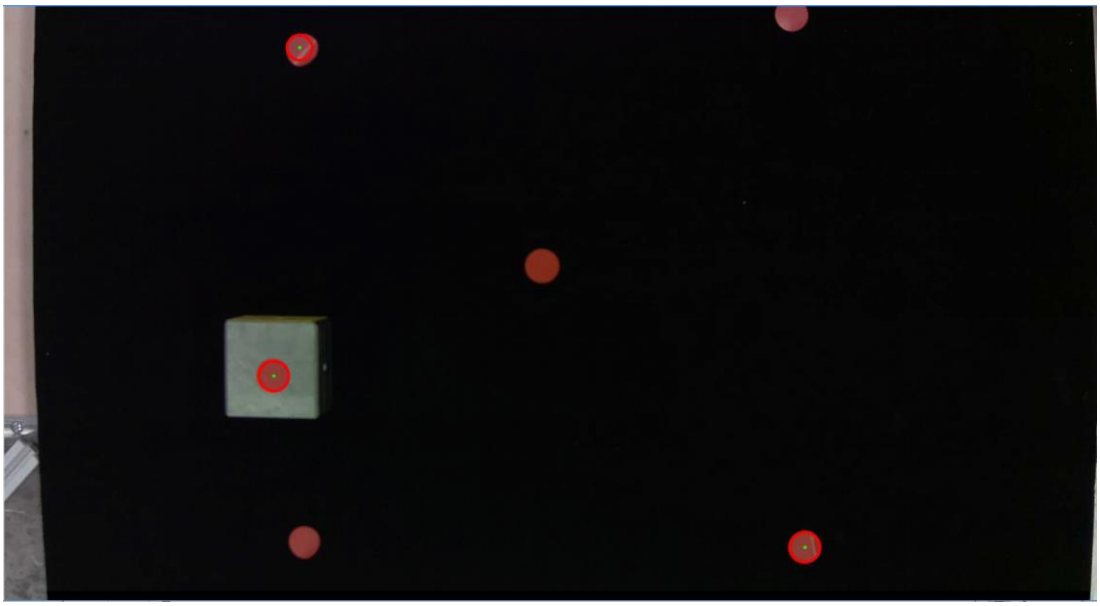


Abbildung 2-11 - Arducam mit 6mm Objektiv aus 1000 mm Höhe [20]

In der finalen Version der Kamera soll die Kamera Objekte für den Roboter erkennen. Damit der Roboter zwischen Kamera und Objekt genügend Bewegungsfreiheit hat ist es also wünschenswert den größtmöglichen Abstand zu verwenden. Somit liegt die Wahl des 6 mm Objektives sehr nah.

2.1.2.4 Verbindungsleitung

Um Kamera und Controller trennen zu können muss das CSI Flachbandkabel verlängert werden. Das Flachbandkabel ist jedoch von der Länge beschränkt und wäre auch aufgrund der fehlenden Schirmung sehr Störempfindlich. Deshalb wird ein CSI Kamera Extension, mittels HDMI Kabel, eingesetzt um eine größere Distanz zu realisieren. Das HDMI Kabel hat ausreichende Schirmung um für Störfreies Bild zu übertragen.

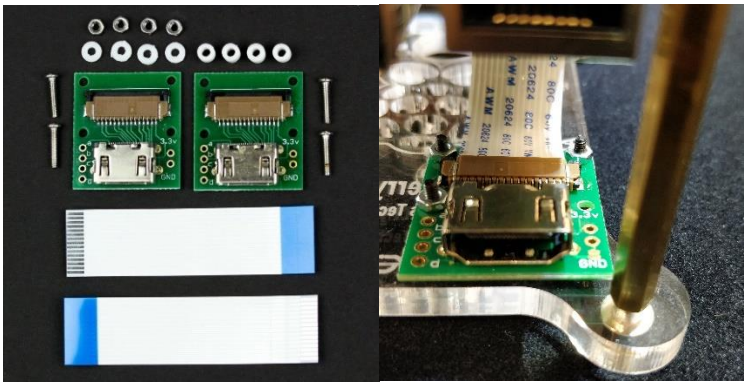


Abbildung 2-12 - CSI Kamera Extender[21]

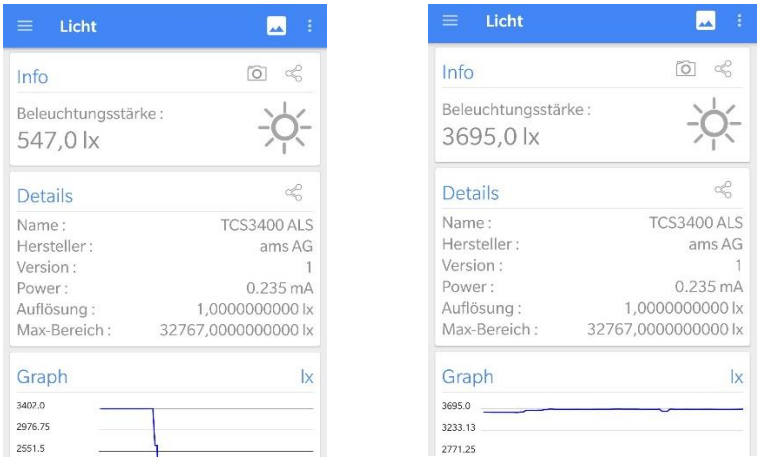
[20] Quelle: Eigene Aufnahme

[21] Pimoroni, „Raspberry Pi Camera HDMI Cable Extension — Pimoroni“, zugegriffen 7. Juni 2018, https://shop.pimoroni.de/products/pi-camera-hdmi-cable-extension.

2.1.2.5 Beleuchtung

Als Beleuchtung werden 2 x 18W LED Scheinwerfer verwendet, die mir einer 19V Spannungsversorgung betrieben werden. Die LED Scheinwerfer sind auf dem KFZ Zubehör und somit kein Profizubehör. Es lässt sich die nicht Helligkeit einstellen oder ein Blitzlicht erzeugen. Es erzeugt aber eine deutlich stärkere Beleuchtungsstärke.

Abbildung 2-13 – Beleuchtung[22]



Vorher Nachher

Abbildung 2-14 - Beleuchtungsstärke[23]

[22] Quelle: Eigene Aufnahme

[23] Quelle: Eigene Aufnahme von der Android Software „Sensoren Multitool“ v 1.3.0 vom Autor: weRed Software

Damit die Objekte gut erkannt werden sollte der Hintergrund einen homogenen Kontrast bieten. Dafür wurde als Untergrund eine Speerholzplatte mit schwarzem Teppich verwendet.

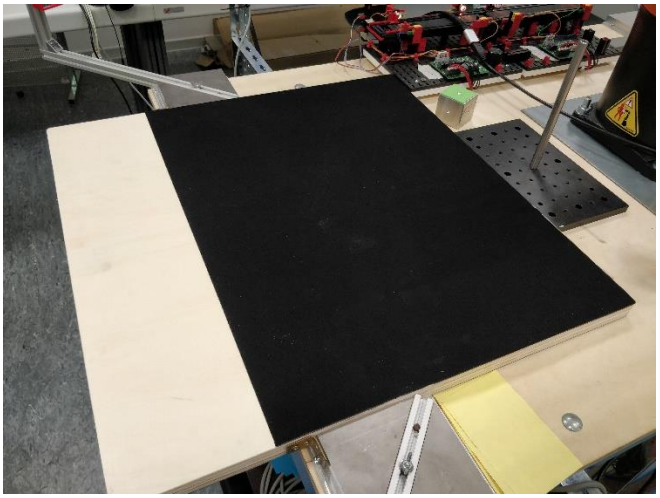


Abbildung 2-15 - Kamera Hintergrund[24]

2.1.2.6 Gehäuse

Für das Gehäuse der Kameramodule wurden Spelsberg Verbindungsdosen verwendet. Jede Kamera wird in einem Deckel fest verschraubt. In die Verbindungsdose selbst wird der HDMI Extender verschraubt und ermöglicht somit das wechseln der Objektive nur durch austauschen des Deckels und Verbindung der Kamera mit dem HDMI Extender mit einem kurzen Stück Flachbandkabel.

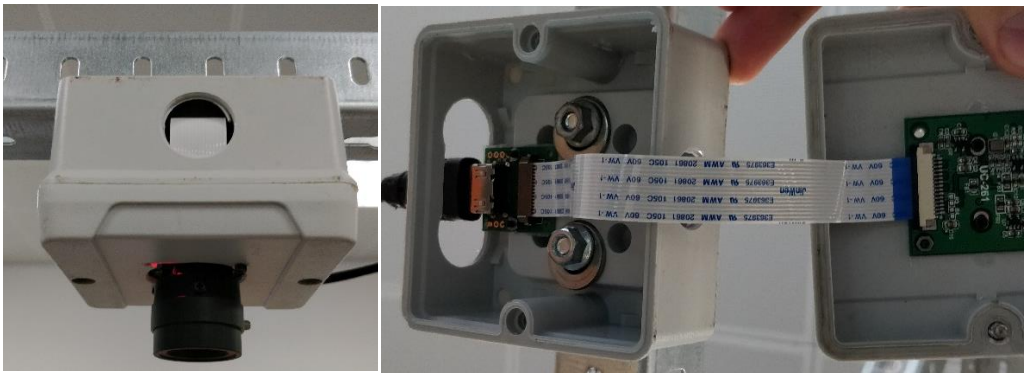


Abbildung 2-16 - Kamera Gehäuse[25]

[24] Quelle: Eigene Aufnahme
[25] Quelle: Eigene Aufnahme

2.1.3 EtherCat

Der EtherCat Bus ist ein Master / Slave Bussystem der eine Echtzeitfähige Datenübertragung ermöglicht. Dafür ist es jedoch auch notwendig das alle Teilnehme störungsfrei laufen und eine feste Zykluszeit innerhalb des Bussystems einhalten. Der Master erzeugt Datenframes und diese müssen von Slave zu Slave weitergereicht werden. Der Slave muss jedes Datenpacket aufnehmen und weitergeben damit der Bus nicht in Störung geht. Somit ist es sinnvoll diesen Prozess auszulagern in einen Mikrocontroller. Das EtherCat Bussystem kann prinzipiell auf jedem PC mit Netzwerkschnittstelle (RJ45) realisiert werden. Jedoch ist für die Echtzeit Funktion einen Intel Chipsatz notwendig (zu mindestens in Kombination mit TwinCat). Außerdem sollte ein EtherCat Slave 2 Netzwerkanschlüsse haben und einen weiteren Teilnehmer in Reihe zu schließen.

Raspberry PI bietet somit nicht die besten Voraussetzungen, jedoch gibt es von der Firma Hilscher ein passendes Shield für den Raspberry PI. Dieser wird einfach auf die GPIO Verbindung vom dem Raspberry PI gesetzt. Raspberry PI und EtherCat Shield kommunizieren später über die SPI Schnittstelle. Die Umsetzung von diesem Modul wird in der Bachelorhtesis behandelt.

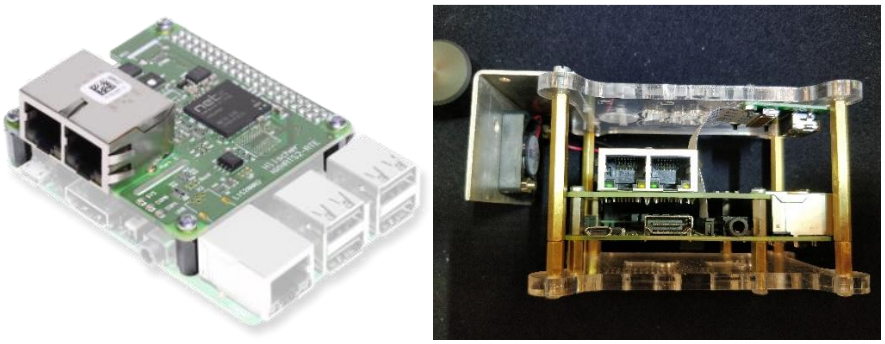


Abbildung 2-17 – Hilscher EtherCat Shield NHAT 52-RE[26]

Durch das Aufstecken des EhterCat Shields sind auch alle anderen unbenutzten GPIO Kontakte nicht mehr nutzbar. Deshalb wurde noch eine PiFace Adapter eingelötet damit zukünftig auch die GPIOs genutzt werden können.

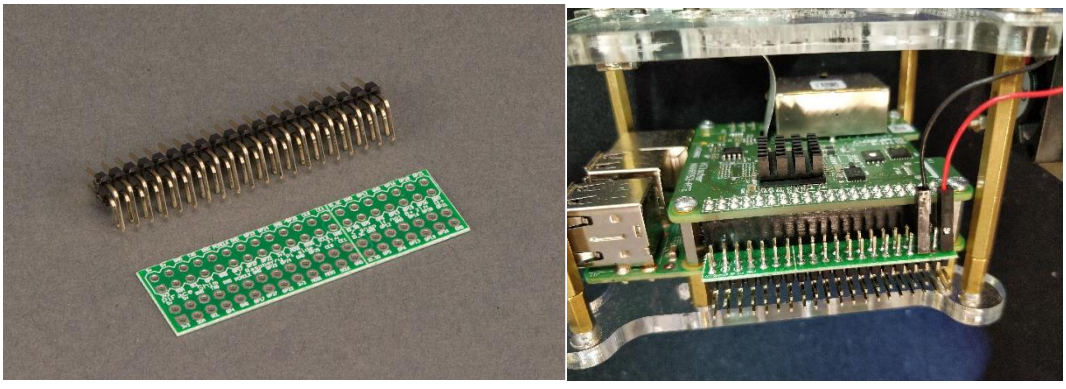


Abbildung 2-18 - PiFace GPIO Shim[27]

[26] Hilscher, „NetHAT“, netIOT, 16. August 2017, www.netiot.com.

[27] Kiwi Electronics, „PiFace GPIO Shim für Raspberry Pi“, Kiwi Electronics, zugegriffen 7. Juni 2018, https://www.kiwi-electronics.nl/piface-gpio-shim.

2.1.4 Gestell

Das Gestell wird aus handelsüblichen Winkel Profilen aus dem Baumarkt zusammengeschraubt. Dabei musste die Breite des Gestells so gewählt werden, dass der Roboter Bewegungsradius zur Seite nicht beschränkt ist.

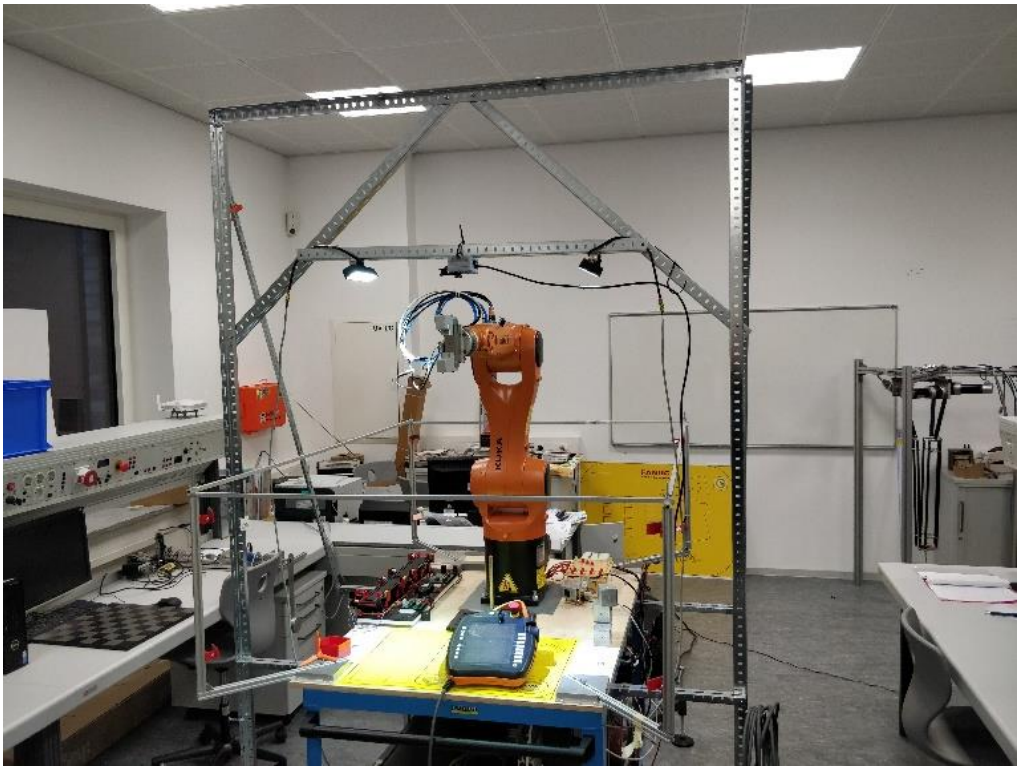


Abbildung 2-19 – Kameragestell[28]

[28] Quelle: Eigene Aufnahme

2.2 Hardwareübersicht

Aufgrund der Randbedingungen kommen diverse Systeme in Frage. In dieser Ausarbeitung wurden jedoch folgende Komponenten ausgewählt:

Systemkomponente	Bauteil	Preis
Controller	Raspberry PI Model 3 B	29,90 €
	Lüfter und Kühlkörper	5,99 €
	Gehäuse	10,29€
	32 GB microSD Karte	14,99€
	USB Ladegerät	11,99 €
	Mirco USB Kabel	5,00 €
	Abstandshalter M3 x 45 mm x 51 mm	4,65 €
	82,81 €	
Kamera	Raspberry Pi Kamera V2 8MP NoIR	29,90 €
	Raspberry Pi Kamera V2 8MP	22,70 €
	Arducam OV5647 CS 5MP LS-2716 4mm	15,00 €
	Arducam OV5647 CS 5MP 0612-3MP-A 6mm	15,00 €
	Raspberry Pi Camera HDMI Cable Extension	21,00 €
	Raspberry Pi Camera Halter	3,50 €
	HDMI Kabel	8,49 €
	LED Beleuchtung	19,29 €
	19 V Netzteil	15,00 €
	3 x Spelsberg Verbindungsdosen	12,00 €
	Sperrholz Platte 600x600	15,70 €
	Klebefolie Velour Schwarz	6,70 €
	184,28 €	
EtherCat	netHAT 'NHAT 52-RE'	39,90 €
	PiFace GPIO Shim	5,95 €
	45,85 €	
Gestell	5 x Stahl Winkel 35,5 x 35,5 x 2500	37,75 €
	2 x Stahl Winkel 35,5 x 35,5 x 1000	6,00 €
	3 x Stahl U Winkel 35,5 x 35,5 x 1000	16,35 €
	2 x Stahl Winkel 24 x 24 x 2500	8,70 €
	50 x M6 x 20 Schrauben	10,00 €
	78,80 €	
391,74 €		

Tabelle 2-1 – Systemkomponenten

2.3 Auswahl der Software

Das Betriebssystem auf dem Raspberry PI bleibt Raspbian. Zur Realisierung der Objekterkennung müssen jedoch noch zusätzliche Programm installiert werden.

2.3.1 OpenCV

Abbildung 2-20 - OpenCV Logo[29]

OpenCV ist eine Open Source Bibliothek mit Algorithmen für die Bildverarbeitung. Die Bibliothek kann in den Programmiersprachen C++, Python und Java verwendet werden. Mit der Grundlage dieser Bibliothek ist somit möglich ein Programm zu entwickeln um Fotos zu erstellen, diese Auszuwerten und die gesammelten Daten weiter zu verwenden um sie ggf. an ein anderes Programm zu übergeben oder über eine Schnittstelle an eine andere Steuerung zu übergeben.[30]

Der Vorteil von OpenCV ist das die Open Source Lizensierung eine kostenlose Nutzung auch für den kommerzielleren Gebrauch zulässt. Desweitern sind Bibliotheken immer Sinnvoll um die Softwareentwicklung zu erleichtern, bzw. auch zu optimieren. Bei der Softwareentwicklung möchte man nicht immer das Rad erneut erfinden, sondern auf bestehende Algorithmen\Methoden\Funktionen zugreifen und diese „nur“ nutzen. Ganz nach dem Prinzip der Datenkapselung. Man muss wissen wie man es nutzt, jedoch muss man nicht wissen wie es intern funktioniert.

[29] OpenCV, „OpenCV library“, zugegriffen 8. Juni 2018, https://opencv.org/.
[30] Vgl. OpenCV.

2.3.2 CODESYS

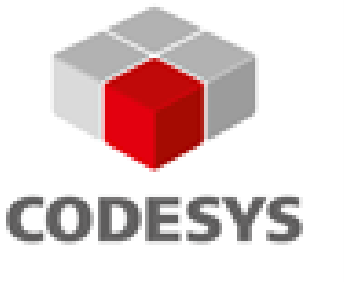


Abbildung 2-21 - CODESYS Logo[31]

CODESYS ist eine Steuerung die auf verschiedenen Software-Plattform betrieben werden kann. So ist zum Beispiel möglich auf einem Windows Rechner eine Software SPS (Speicherprogrammierbare Steuerung) zu betreiben. Es ist aber auch möglich die gleiche Steuerung auf einem Raspberry PI zu betreiben. Möglich ist dies durch die Runtime von CODESYS die es für verschiedene Plattformen gibt. Mit CODESYS können auch Visualisierungssysteme und Motion Control System erstellt werden. [32]

Das System ist sehr Vielseitig und kann mit verschiedenen Programmiersprachen nach IEC 61131-3 programmiert werden, wie zum Beispiel FBS/KOP/AWL aber auch ST[33]

Die Lizensierung richtet sich dabei an die verwendete Runtime. So ist zum Beispiel die Nutzung der Entwicklungsumgebung kostenlos. Jedoch muss der Controller mit einer passenden Lizenz ausgestattet werden.[34]

Eine Windows Lizenz liegt zum Beispiel bei 420 Euro, die vom Raspberry Pi liegt bei gerade mal 50 €. Desweitern ist es möglich die Systeme auch ohne die gekaufte Lizenz zu verwenden, jedoch schalten sich die Systeme nach 2 Stunden hab und man muss sie neustarten. Der Funktionsumfang ist jedoch voll gegeben wodurch die Nutzung als Entwicklungsumgebung bzw. Schulungsstation völlig ausreichend ist.[35]

CODESYS biete zu der Runtime für Raspberry Pi auch direkt eine Weboberfläche an, mit der eine Visualisierung für die Steuerung erstellt werden kann. Als kleinen Ausblick für andere Realisierung Möglichkeiten könnten man damit den Raspberry Pi als Hausautomatisierungssteuerung verwenden und die Bedienung vom Smartphone über einen normalen Browser ermöglichen.

---

[31] CODESYS, „Das System“, zugegriffen 8. Juni 2018, https://de.codesys.com//das-system.html.

[32] CODESYS.

[33] Vgl. Matthias Seitz, *Speicherprogrammierbare Steuerungen für die Fabrik- und Prozessautomation*, 4., überarbeitete und erweiterte Auflage (München: Fachbuchverlag Leipzig im Carl Hanser Verlag, 2015), 69.

[34] Vgl. CODESYS, „Lizenzierung“, zugegriffen 8. Juni 2018, https://de.codesys.com//das-system/lizenzierung.html.

[35] Vgl. CODESYS, „CODESYS Store - Systeme“, zugegriffen 8. Juni 2018, https://store.codesys.com/systeme.html?p=2.

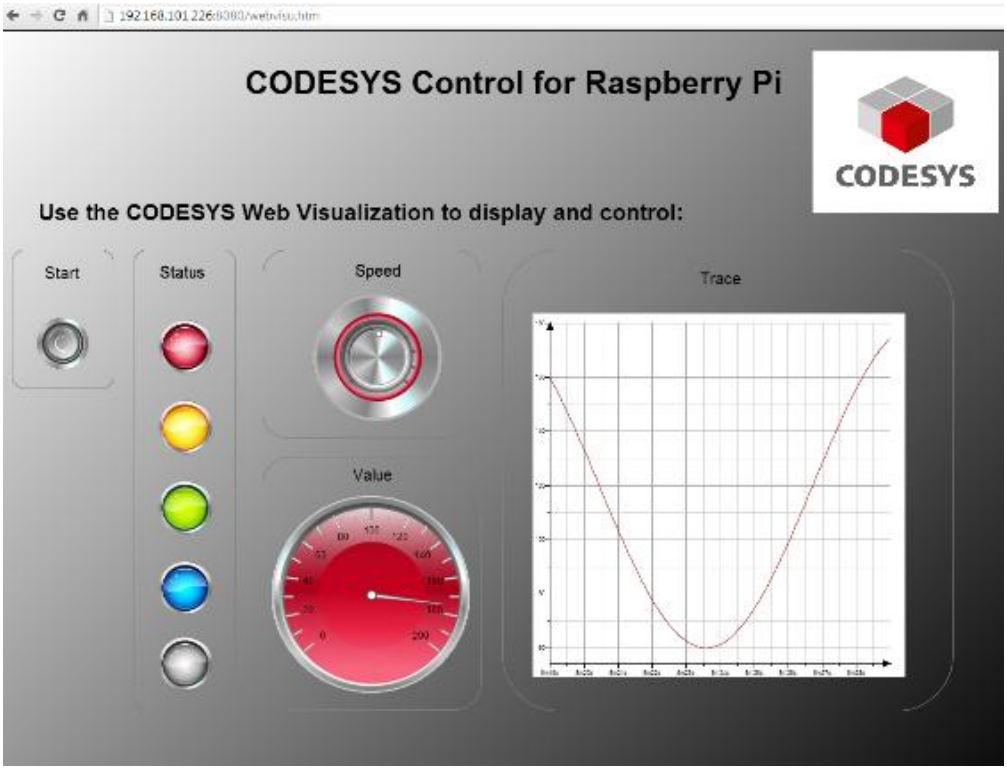


Abbildung 2-22 - WebVisualisierung Codesys Raspberry PI[36]

Desweitern biete CODESYS auch ein passende AddOn für OpenCV für die Raspberry PI Runtime an wodurch das System auch für Kameraapplikation interessant wird.[37] Mit der Entwicklungsumgebung kann somit komfortable vom eigenen Rechner ein Projekt erstellt werden und auf dem Raspberry ausgeführt werden.

[36] CODESYS, „CODESYS Store - CODESYS Control for Raspberry Pi SL“, zugegriffen 8. Juni 2018, https://store.codesys.com/systeme/codesys-control-for-raspberry-pi-sl.html#1.

[37] Vgl. CODESYS, „CODESYS Store - OpenCV for Raspberry Pi“, zugegriffen 8. Juni 2018, https://store.codesys.com/opencv-for-raspberry-pi.html.

3 Implementierung

3.1 Grundinstallation

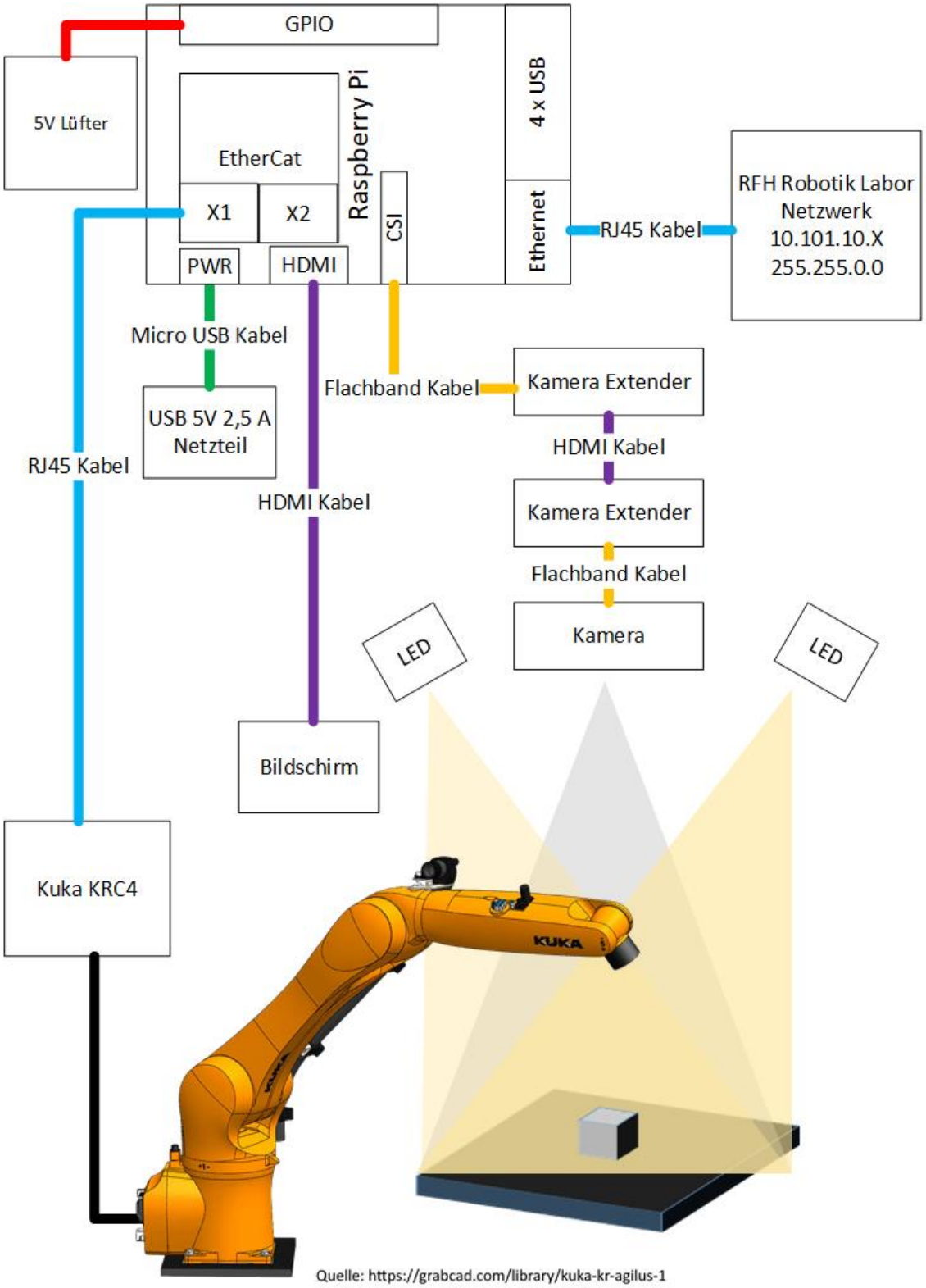


Abbildung 3-1 – Aufbau[38]

[38] Quelle: Erstellt mit Microsoft Visio

3.2 Betriebssystem

Für das Betriebssystem auf dem Raspberry bietet sich am besten die offizielle Linux Distribution Raspbian an. Ein passendes Image kann auf der folgenden Seite heruntergeladen werden. https://www.raspberrypi.org/downloads/raspbian/

Das Betriebssystem wird auf eine Micro SD Karte kopiert und danach lässt sich der Raspberry auch schon nutzen. Das Besondere daran ist, dass die SD Karte schnell getauscht werden kann um das System mit einer anderen Software Variante laufen zu lassen. Die aktuellste Version von Raspbian ist derzeit „Stretch“ und dieser wird auch von der neusten CODESYS Version unterstützt, leider werden aber für das Addon OpenCV Pakete benötigt die derzeit nur in der alten Version „Jessie“ laufen. Im weiteren Verlauf wird auch das neue OS Stretch somit werden 2 SD Karten vorbereitet. Die SD Karte muss min. 8 GB groß sein damit das Image auf die Karte passt. Für den Transfer des Images eignet sich am besten die Software „Win32 Disk Imager“.

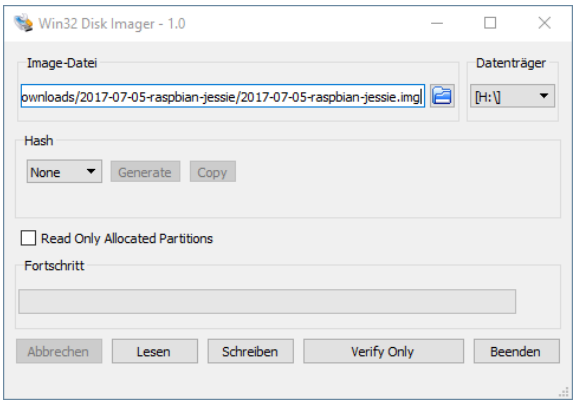


Abbildung 3-2 - Win32Disk[39]

Nach dem erfolgreichen Transfair ist unter Windows nur noch das Boot Laufwerk der SD Karte sehen. In diesem können noch Modifikationen durchführt werden

3.2.1 SSH Zugriff erlauben

Dazu eine leere Datei mit dem Dateinamen „ssh“ ohne Dateitypangabe im Bootlaufwerk ablegen[40]

[39] Quelle: Eigene Aufnahme

[40] Vgl. Raspberry Tips, „SSH am Raspberry Pi aktivieren – Neue Methode mit Raspbian 25.11.2016“, *raspberry.tips* (blog), 23. Dezember 2016, https://raspberry.tips/raspberrypi-tutorials/ssh-am-raspberry-pi-aktivieren-neue-methode-mit-raspbian-25-11-2016.

3.2.2 WLAN einrichten

Eine Datei mit dem Namen „wpa_supplicant.conf“ im Bootlaufwerk ablegen

```
# Datei wpa_supplicant.conf in der Boot-Partition (Raspbian Jessie)
network={
       ssid="wlan-bezeichnung"
       psk="passwort"
       key_mgmt=WPA-PSK
}
# Datei wpa_supplicant.conf in der Boot-Partition (Raspbian Stretch)
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
       ssid="wlan-bezeichnung"
       psk="passwort"
       key_mgmt=WPA-PSK
}
```

Code 3-1 - Wlanconfig[41]

3.2.3 Systemstart

Die SD Karte kann nun in den Raspberry Pi und dieser danach mit der USB Spannungsversorgung verbunden werden. Beim Booten des Systems wird automatisch eine WLAN Verbindung aufgebaut und der SSH Zugriff erlaubt.

3.2.4 Verbindung aufbauen

Der Raspberry PI bootet in einer eigenen GUI und könnte mit einem angeschlossen HDMI Bildschirm sowie Maus und Tastatur bedient werden. Jedoch ist es oft einfacher nur eine SSH Verbindung aufzubauen. Dafür kann die Software PuTTY verwendet werden.

Benutzername „pi“ Passwort „raspberry“.

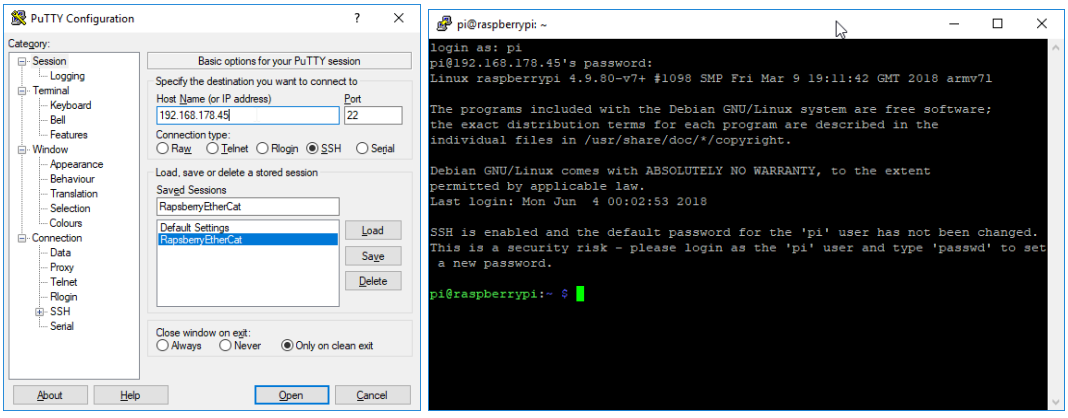


Abbildung 3-3 – PuTTY[42]

---

[41] Vgl. Michael Kofler, „WLAN schon vor der Inbetriebnahme konfigurieren | pi-buch.info“, zugegriffen 8. Juni 2018, https://pi-buch.info/wlan-schon-vor-der-installation-konfigurieren/.

[42] Quelle: Eigene Aufnahme

3.2.5 Grundkonfiguration

Es müssen nun ein paar Grundeinstellungen vorgenommen werden. Dazu wird das Konfigurations-menü von Rasbian aufgerufenen.

```
sudo raspi-config
```

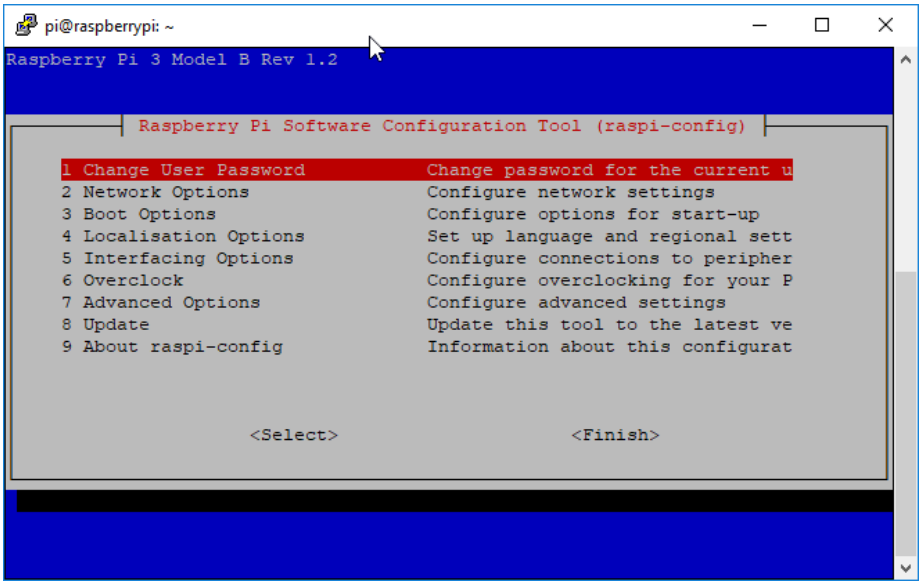


Abbildung 3-4 - Raspi-config[43]

In diesem Menü sollten das Passwort des Benutzers „pi“ geändert werden und die „Localisation Option“ angepasst werden.

Außerdem muss unter „Interfacing Option“ die Optionen „Camera“ und „VNC“ aktiviert werde. Unter „Advanced Option“ sollte noch die Funktion „Expand Filesystem“ gestartet werden damit Raspbian die gesamte SD Karte zur Verfügung hat.

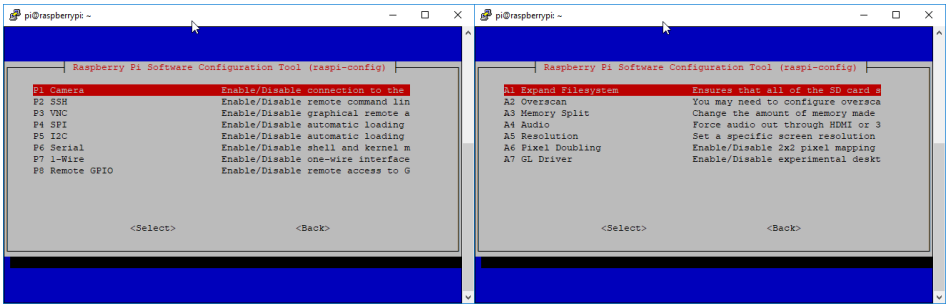


Abbildung 3-5 - Raspi-Config[44]

[43] Quelle: Eigene Aufnahme
[44] Quelle: Eigene Aufnahme

3.2.6 VNC

Durch das aktiveren der Option VNC kann man nun auch über einen passenden VNC Viewer (in diesem Fall RealVNCViewer) direkt eine Verbindung auf die GUI von Raspbian aktiveren.

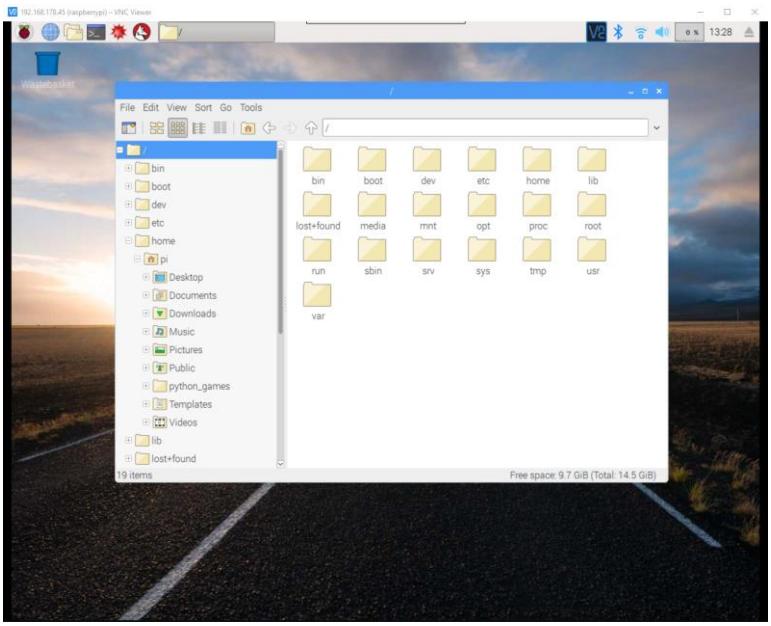


Abbildung 3-6 – VNC[45]

3.2.7 Samba

Um Daten mit dem Raspberry PI auszutauschen kann man einen USB Stick verwenden oder man macht sich mal wieder Netzwerkverbindung zu nutze. Der einfachste Weg um zwischen Windows und Raspberry Daten auszutauschen ist „Samba“.

Für die Installation müssen die folgenden Schritte durchgeführt werden:[46]

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get update
sudo apt-get install samba samba-common smbclient
```

Danach muss das Config File angepasst werden.

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf_alt
sudo nano /etc/samba/smb.conf

[global]
workgroup = WORKGROUP
security = user
encrypt passwords = yes
[SambaRoot]
comment = Samba-Root-Freigabe
path = /
read only = no

Strg +x zum Beenden und speichern
```

[45] Quelle: Eigene Aufnahme

[46] Vgl. Patrick Schnabel, „Samba-Freigabe auf dem Raspberry Pi einrichten“, zugegriffen 8. Juni 2018, https://www.elektronik-kompendium.de/sites/raspberry-pi/2007071.htm.

Der Netzwerkzugriff wird nur erlaubt, wenn Benutzername und Passwort eingegeben werden. Der Benutzer PI ist in Samba schon hiterlegt worden jedoch ist hier nicht automatisch das gleiche Passwort. Es empfiehlt sich aber hier das gleiche zu verwenden:

```
sudo smbpasswd -a pi
```

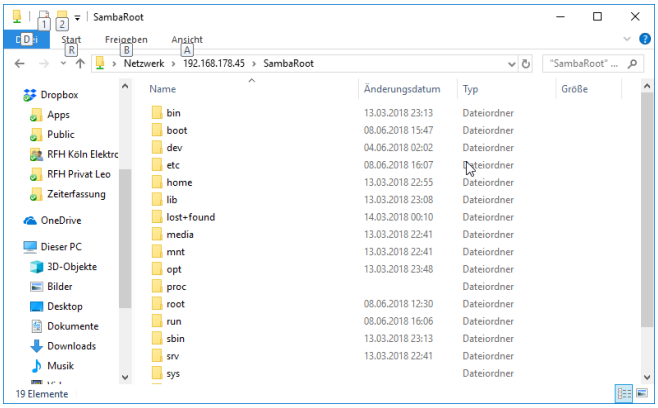


Abbildung 3-7 – Samba[47]

3.2.8 Kamera

Um zu prüfen ob die Kamera richtig angeschlossen ist kann auch ohne OpenCV und ohne ein Programm ein Bild zur Testzwecken erstellt werden:

Mit dem Befehl „raspistill“ wird das Kameramodul angesprochen und ein Foto erstellt. Dieser Befehl lässt sich über PuTTY oder das Terminal Fenster direkt auf Raspbian starten.[48]

```
raspistill -o ~/bild.jpg
```

Nun wird ein Bild erzeugt und im Homeverzeichnis (~ = /home/pi/) unter dem Namen Bild.jpg abgespeichert. Nach Absetzen des Befehls wird auf der GUI ein Vorschaubild angezeigt um die Kamera passend auszurichten. Diese Bild wird aber leider nicht über VNC übertragen. Sodass es nur möglich ist sich später das fertige Bild anzuzeigen. Mit den passenden Parametern kann man das Kamera Bild auch noch auf die Gegebenheiten anpassen.

```
raspistill -ISO 800 -ex auto -awb auto -o ~/bild.jpg
```

Mehr dazu kann unter der angegebenen Quelle nachgelesen werden.

---

[47] Quelle: Eigene Aufnahme

[48] Vgl. „Raspberry Pi Camera Module - Raspberry Pi Documentation“, zugegriffen 8. Juni 2018, https://www.raspberrypi.org/documentation/raspbian/applications/camera.md.

3.3 CodeSys

Für die Installation von Codesys in Kombination mit OpenCV ist es notwendig Rasbian Jessie zu nutzen.

Zunächst wird die Entwicklungsumgebung auf einem Windowsrechner vorbereitet. Dazu muss die Software „CODESYS Development System V3“ installiert werden. https://store.codesys.com/codesys.html

Nach der Installation müssen noch zusätzliche Packet runtergeladen werden und installiert werden:

- https://store.codesys.com/codesys-control-for-raspberry-pi-sl.html?store=default
- https://store.codesys.com/opencv-for-raspberry-pi.html
- https://store.codesys.com/codesys-softmotion-sl.html?store=default

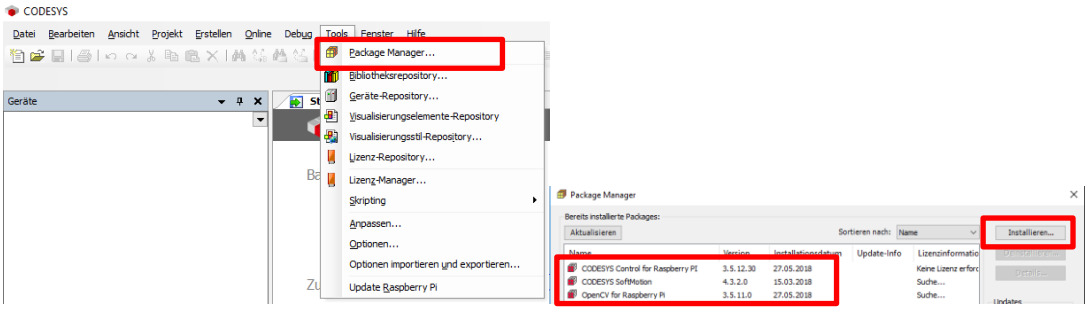


Abbildung 3-8 - CODESYS Package Manager[49]

Auf dem Raspberry PI muss nun die OpenCV Bibliothek installiert werden. Dazu müssen die folgenden Schritte:[50]

```
sudo apt-get install libopencv-dev=2.4.9.1+dfsg-1+deb8u1
```

Außerdem muss noch ein temporäres Filesystem eingerichtet werden.

```
sudo -s
nano /etc/fstab

tmp         /tmp            tmpfs           defaults        0       0

shutdown -r -t 0
```

[49] Quelle: Eigene Aufnahme

[50] Vgl. Robert Lang, „OpenCV for Raspberry Pi“, o. J., zugegriffen 7. Juni 2018.

Anschließen kann die Installation von CODESYS auf dem Raspberry gestartet werden. Das geht sehr komfortable über die CODESYS Entwicklungsumgebung.

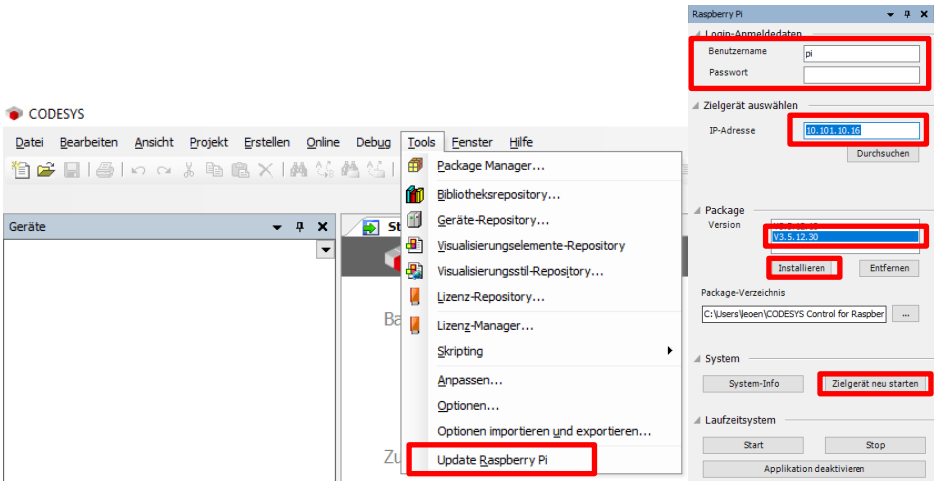


Abbildung 3-9 – CODESYS Raspberry Installation[51]

Mit dem Packet „OpenCV for Raspberry PI“ wird im Home Profil des Windows Rechners auch ein Verzeichnis erstellt, mit Dokumentationen und einem Installations-File für die Raspberry PI und einem Beispiel Projekt. Öffnen Sie diese und verbinden sich mit dem Raspberry PI

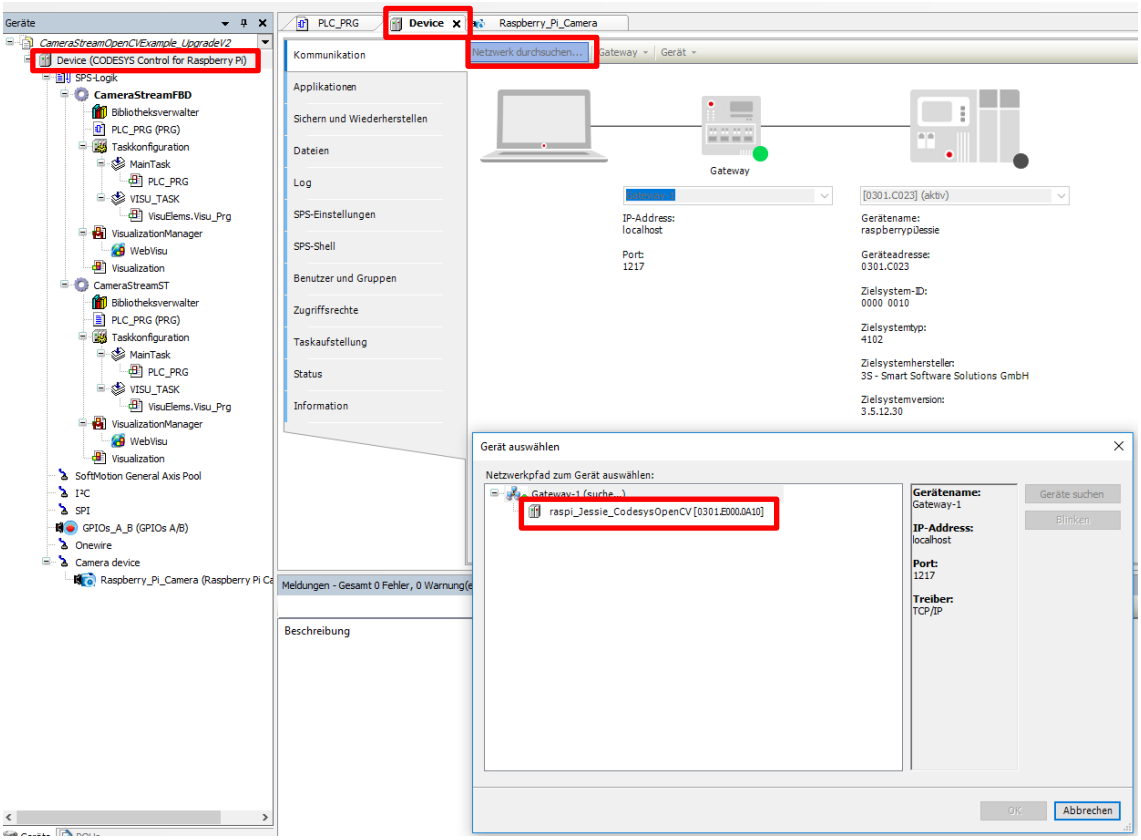


Abbildung 3-10 – CODESYS Raspberry einbinden[52]

[51] Quelle: Eigene Aufnahme

[52] Quelle: Eigene Aufnahme

27

Übertragen Sie das zusätzlich enthaltenden Installations-File auf den Raspberry PI

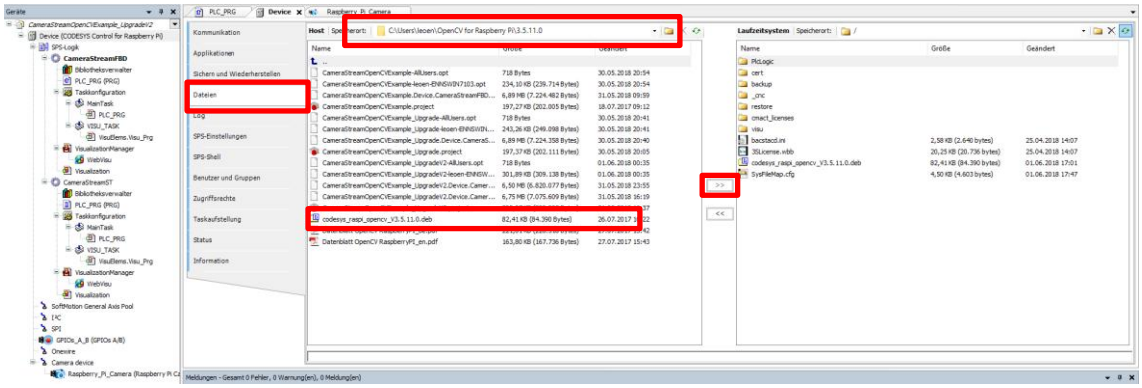


Abbildung 3-11 – CODESYS Datentransfair zu Raspberry[53]

Und starten die Installation mit folgendem Befehl im Terminal.

```
dpkg -i codesys_raspi_opencv_V3.5.11.0.deb
```

Im letzten Schritt kann nur das Projekt auf den Raspberry PI übertragen werden.

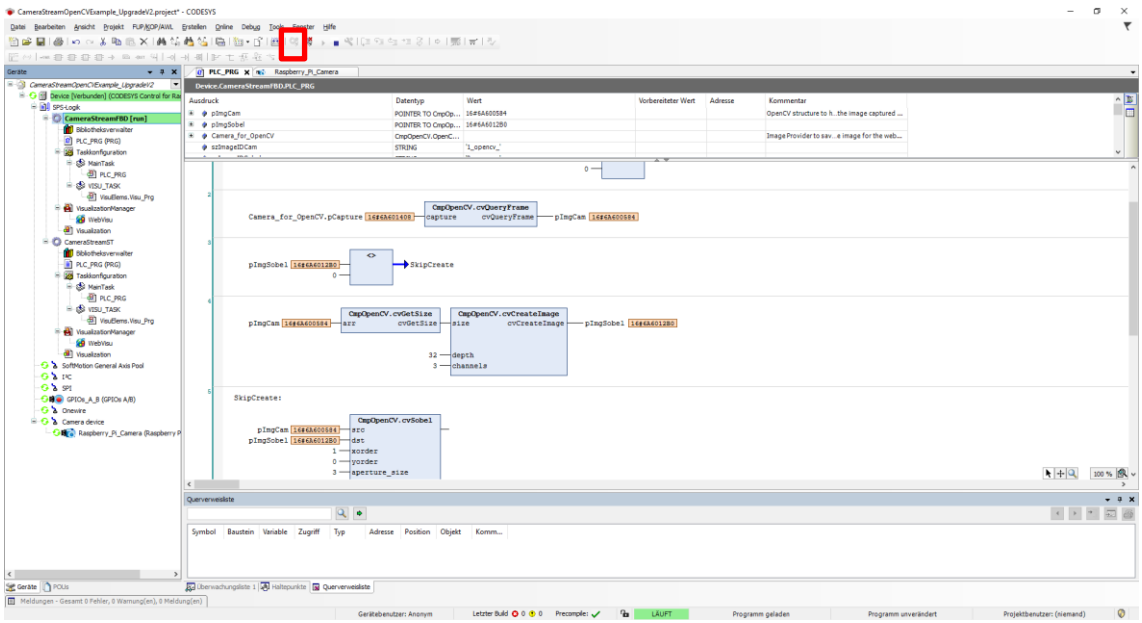


Abbildung 3-12 – CODESYS Projekt übertragen[54]

[53] Quelle: Eigene Aufnahme

[54] Quelle: Eigene Aufnahme

In dem Beispiel Projekt wird das Bilder der Kamera mit der Funktion „cvSobel“ konvertiert um Kanten deutlicher zu erkennen. Original Bild und konvertiertes Bild werden auf einem Webinterface ausgegeben

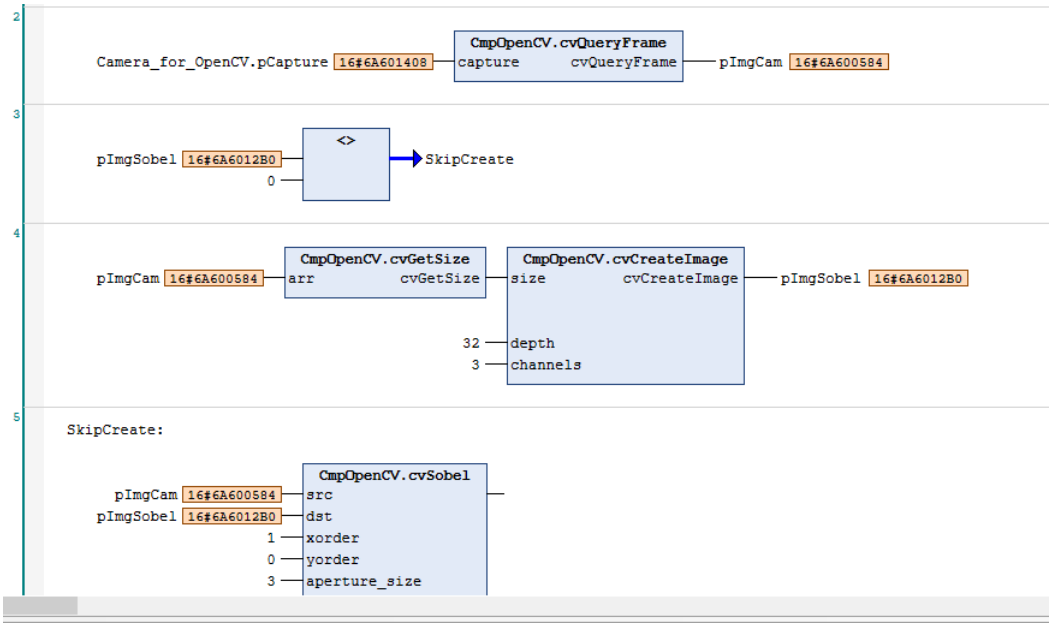


Abbildung 3-13 – CODESYS OpenCV Programmbeispiel[55]

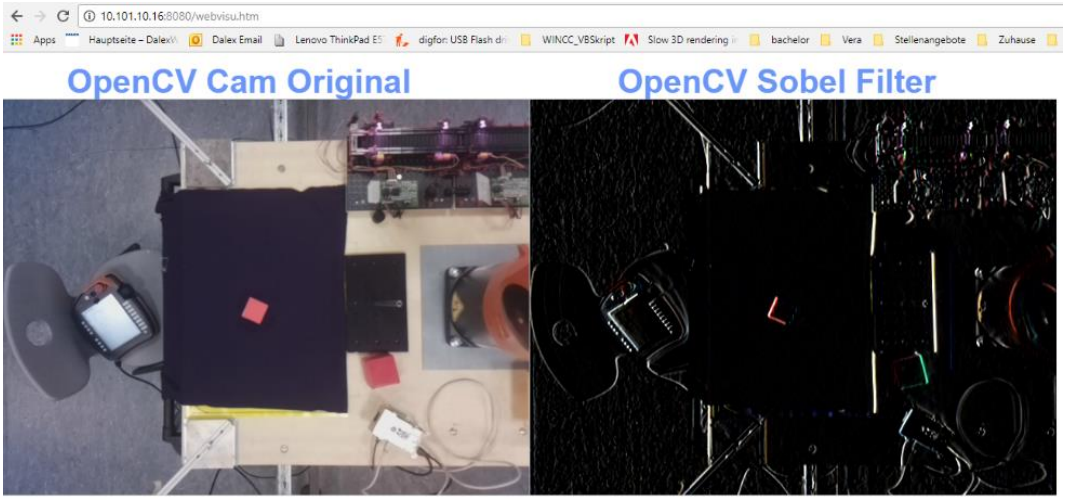


Abbildung 3-14 – CODESYS Raspbery Webvisu[56]

[55] Quelle: Eigene Aufnahme
[56] Quelle: Eigene Aufnahme

In dem PLC_PRG könnte man auch auf die GPIO des Raspberry Pis zugreifen.

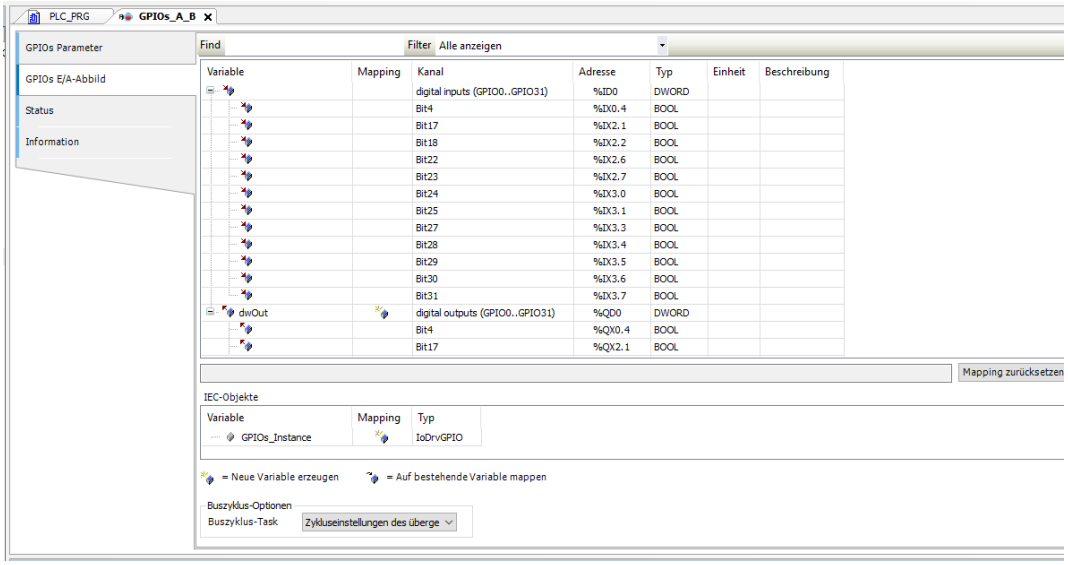


Abbildung 3-15 – CODESYS Raspberry GPIO konfiguration[57]

[57] Quelle: Eigene Aufnahme

3.4 OpenCV

Die CODESYS Welt bringt viele Vorteile bedarf aber auch einiges an KnowHow um alle Funktionen zu realisieren.

An Dieser Stelle musste ich leider einen anderen Weg einschlagen, weil ich nicht mehr genügend Zeit hatte um mich in CODESYS einzuarbeiten.

Somit werden wir nun das System auf der 2 SD Karte nochmal neu aufsetzen aber mit der neueste Rasbian Version „Stretch“. Der Ablauf ist der gleiche und die Schritte von 3.1 können genauso durchgeführt werden.

Das Grundgerüst für das System wird die Programmiersprache Python 3 sein.

Eine gute Grundlage für diese Programmiersprache bietet das das Buch Raspberry Pi Das umfassende Handbuch von Michael Kofler, Charly Kühnast und Christoph Scherbeck.[58]

Python 3 ist auf dem Raspberry schon vorinstalliert und ist auch das Standard Programmiesprache auf dem Raspberry.[59]

Für die Installation von OpenCV sind einige Schritte notwendig die auch sehr zeitintensiv sind. Die notwendigen Schritte sind auf pyimageserach gut beschrieben.[60]

Schritt 1: Grundinstallation

Zusätzlich zu den Schritten von Abschnitt 3.1 sollten nun das Betriebsystem Raspbian ein wenig aufgeräumt werden um Platz einzusparen:

```
sudo apt-get purge wolfram-engine
sudo apt-get purge libreoffice*
sudo apt-get clean
sudo apt-get autoremove
```

Schritt 2: Raspbian Vorbereitung

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get update
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libgtk2.0-dev libgtk-3-dev
sudo apt-get install libatlas-base-dev gfortran
sudo apt-get install python3-dev
```

[58] Vgl. Michael Kofler, *Raspberry Pi das umfassende Handbuch*, 2017, http://nbn-resolving.de/urn:nbn:de:101:1-201708226939.

[59] Vgl. Michael Kofler.

[60] Vgl. Adrian Rosebrock, „Raspbian Stretch: Install OpenCV 3 + Python on Your Raspberry Pi“, *PyImageSearch* (blog), 4. September 2017, https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/.

Schritt 3: OpenCV source Code runterladen

```
wget -O opencv.zip https://github.com/opencv/opencv/archive/3.4.1.zip
unzip opencv.zip
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/ar-
chive/3.4.1.zip
unzip opencv_contrib.zip
```

Schritt 4: Python 3 Vorbereiten

```
sudo apt-get install python3-pip
sudo easy_install pip
sudo pip install virtualenv virtualenvwrapper
sudo pip install --upgrade virtualenvwrapper
sudo rm -rf ~/.cache/pip
sudo nano ~/.profile

# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh

source ~/.profile
mkvirtualenv cv -p python3

reboot
```

Schritt 5: Numpy installieren

```
source ~/.profile
workon cv
pip install numpy
```

Schritt 6: OpenCV kompilieren

Beim letzten Schritt ist es wichtig, das der Raspberry PI ausreichen gekühlt ist, weil die Kompilierung etwa 2 Stunden dauert und sehr Rechenintensiv ist.

```
cd ~/opencv-3.4.1
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
    -D BUILD_EXAMPLES=ON ..

sudo nano /etc/dphys-swapfile
CONF_SWAPSIZE=1024

sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start

make -j4
```

Schritt 7: Installation von OpenCV

```
sudo make install
sudo ldconfig
cd /usr/local/lib/python3.5/site-packages/
ls -l
```

Hier sollte nun folgende Ausgabe stehen

```
total 4500
-rw-r--r-- 1 root staff 4604912 Jun  1 23:40 cv2.cpython-35m-arm-linux-gnueabihf.so
```

```
sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so
cd ~/.virtualenvs/cv/lib/python3.5/site-packages/
ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so
```

Schritt 8: Testen

```
source ~/.profile
workon cv
python
```

Wenn diese Ausgabe erfolgt sollte das System Einsatzbereit sein.

```
>>> import cv2
>>> cv2.__version__
'3.4.1'
>>>
```

Jetzt kann das System noch bereinigt werden.

```
rm -rf opencv-3.4.1 opencv_contrib-3.4.2

sudo nano /etc/dphys-swapfile
CONF_SWAPSIZE=100
```

3.5 Kamera aktivieren

Zunächst müssen die Firmware Stände aktualisiert werden:[61]

```
sudo rpi-update
```

Danach einmal neustarten.

```
sudo modprobe bcm2835-v4l2
```

Zur Kontrolle müsste nur folgende Ausgabe erfolgen.

```
ls -l /dev/video0
crw-rw----+ 1 root video 81, 0 Jun  9 00:59 /dev/video0
```

[61] Vgl.DougieLawson, „Raspberry Pi Forums“, zugegriffen 8. Juni 2018, https://www.raspberrypi.org/forums/viewtopic.php?t=68247.

3.6 Programm Vision Detection

Als Grundlage für das Programm wurde das Beispiel Projekt „OpenCV_test_3.py“[62] von Chris Dahms, verwendet. Es wurden auch noch andere Quellen mit einbezogen.[63][64]

Der Grundaufbau eines Python Programmes sieht wie folgt aus. Besonders wichtig ist der Import der zuvor erstellten Bibliothek cv2

```
1   #OpenCV_RedPoint_Detect.py
2
3   import cv2
4   import numpy as np
5   import os
6
7   ###############################################################################################
    #############
8   def main():


105 ###############################################################################################
    #############
106 if __name__ == "__main__":
107     main()
108
109
```

Abbildung 3-16 - Python Quellcode Rumpf[65]

Nun wird die Verbindung zur Kamera erstellt und die Auflösung definiert. Falls es bei diesen Schritten Fehler geben sollte, werden entsprechende Fehlertexte ausgegeben.

```
9       # erzeugt ein VideoCapture Objekt und verbindet es mit der Kamera 0 (/dev/video0)
10      capWebcam = cv2.VideoCapture(0)
11      # Zeigt Standard Auflösung
12      print("default resolution = " + str(capWebcam.get(cv2.CAP_PROP_FRAME_WIDTH)) +
        "x" + str(capWebcam.get(cv2.CAP_PROP_FRAME_HEIGHT)))
13      # Anpassungen der Auslösung auf 620 x 465 passend zum Objektiv
14      capWebcam.set(cv2.CAP_PROP_FRAME_WIDTH, 640.0)
15      capWebcam.set(cv2.CAP_PROP_FRAME_HEIGHT, 480.0)
16      # Anpassung des Belichtungsprogamm
17      capWebcam.set(cv2.CAP_PROP_EXPOSUREPROGRAM,2)
18      # Zeigt neue Auflösung
19      print("updated resolution = " + str(capWebcam.get(cv2.CAP_PROP_FRAME_WIDTH)) +
        "x" + str(capWebcam.get(cv2.CAP_PROP_FRAME_HEIGHT)))
20      # Kontrolle ob die Kamera angeschlossen ist
21      if capWebcam.isOpened() == False:
22          print("error: capWebcam not accessed successfully\n\n" )
23          # Text wird angezeigt bis eine Taste gedrück wird
24          os.system("pause")
25          # Programm endet
26          return
27      # end if
```

Abbildung 3-17 - Python Quellcode Initialisierungslauf

---

[62] Vgl.Chris Dahms, *Contribute to Raspberry_Pi_2_and_OpenCV_3_Tutorial_Part_1 development by creating an account on GitHub*, Python, 2018, https://github.com/MicrocontrollersAndMore/Raspberry_Pi_2_and_OpenCV_3_Tutorial_Part_1.

[63] Vgl.Abid K, *OpenCV2-Python-Tutorials: This repo contains tutorials on OpenCV-Python library using new cv2 interface*, Python, 2018, https://github.com/abidrahmank/OpenCV2-Python-Tutorials.

[64] Vgl.Bernd Klein, „Python-Tutorial: Dateien lesen und schreiben“, zugegriffen 9. Juni 2018, https://www.python-kurs.eu/dateien.php.

[65] Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py

Solange die Kamera funktioniert und nicht mit der ESC-Taste das Programm beendet wird, wird die Schleife immer wieder ausgeführt. Zu Beginn der Schleife wird ein neues Bild eingelesen

```
29      while cv2.waitKey(1) != 27 and capWebcam.isOpened():
30          # Naechstes Bild laden
31          blnFrameReadSuccessfully, imgOriginal = capWebcam.read()
32          # Ins Bild Zoomen
33          # imgOriginal = img[200:1000, 600:1200]
34          # Kontrolle ob das neue Bild korrekt ist
35          if not blnFrameReadSuccessfully or imgOriginal is None:
36              print("error: frame not read from webcam\n")
37              # Text wird angezeigt bis eine Taste gedrück wird
38              os.system("pause")
39              # Programm endet
40              break
41          # end if
```

Abbildung 3-18 - Python Quellcode Whileschleife [66]

Das Bild wird nun in den HSV Farbraum konvertiert und ein neues Bild erstellt mit einer schwarzen Maske die nur den zuvor angegebenen Farbbereich in weiß darstellt.

```
43          # Bild wird in HSV konvertiert
44          # H = 0-360 Grad in Open CV 0 - 179
45          # S = Verschiebung von weiss bis zum Farbton 0 - 100 in OpenCV 0 - 255
46          # V = Verschiebung von schwarz bis zum Farbton 0 - 100 in OpenCV 0 - 255
47          # Rot ->         0 255 255
48          # Orange ->      15 255 255
49          # Gelb ->        30 255 255
50          # Gruen->        60 255 255
51          # Dunkel Gruen->60 255 127
52          # hell gruen->   60 127 255
53          # Magenta    ->  150 255 255
54          imgHSV = cv2.cvtColor(imgOriginal, cv2.COLOR_BGR2HSV)
55
56          imgThreshLow = cv2.inRange(imgHSV, np.array([0, 135, 135]), np.array([15,
            255, 255]))
57          imgThreshHigh = cv2.inRange(imgHSV, np.array([150, 135, 135]),
            np.array([179, 255, 255]))
58
59          imgThresh = cv2.add(imgThreshLow, imgThreshHigh)
60          # weichzeichner
61          imgThresh = cv2.GaussianBlur(imgThresh, (3, 3), 2)
62          # Morphological Transformations
63          imgThresh = cv2.dilate(imgThresh, np.ones((5,5),np.uint8))
64          imgThresh = cv2.erode(imgThresh, np.ones((5,5),np.uint8))
65
```

Abbildung 3-19 - Python Quellcode Farbraumconvertierung[67]

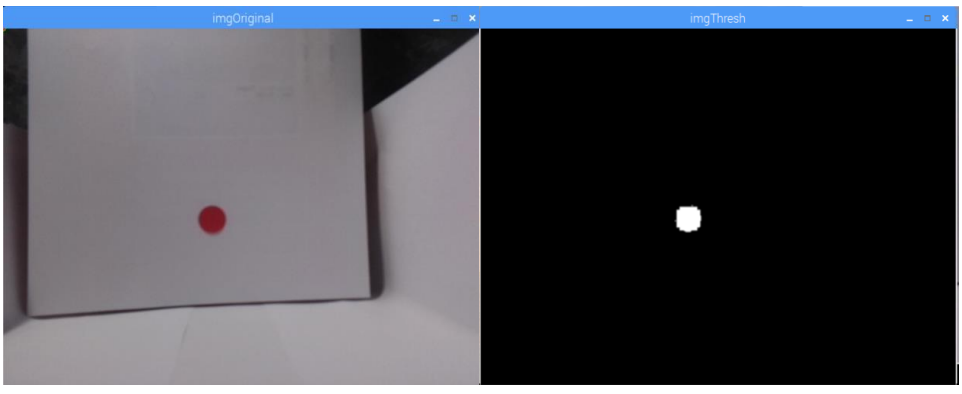


Abbildung 3-20 – Programm Ausgabe[68]

[66] Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py
[67] Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py
[68] Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py

Das gefilterte Bild wird nun auf Kreise durchsucht. Wenn ein Kreis gefunden wurde, werden die x , y Positionen (Pixel) in die Datei Pos.dat gespeichert und im original Bild eingezeichnet.

```
66          intRows, intColumns = imgThresh.shape
67          # fill variable circles with all circles in the processed image
68          circles = cv2.HoughCircles(imgThresh, cv2.HOUGH_GRADIENT, 5, intRows / 4)
69          # this line is necessary to keep program from crashing on next line if no
            circles were found
70          if circles is not None:
71              # for each circle
72              for circle in circles[0]:
73                  # break out x, y, and radius
74                  x, y, radius = circle
75                  os.remove("Pos.dat")
76                  fout=open("Pos.dat","w")
77                  fout.write(str(x)+"\n")
78                  fout.write(str(y)+"\n")
79                  fout.write(str(radius)+"\n")
80                  fout.close()
81                  # print position and radius
82                  print("circle position x = " + str(x) + ", y = " + str(y) + ",
                    radius = " + str(radius))
83                  #print(capWebcam.get(cv2.CAP_PROP_EXPOSURE))
84                  # draw small green circle at center of detected object
85                  cv2.circle(imgOriginal, (x, y), 3, (0, 255, 0), -1)
86                  # draw red circle around the detected object
87                  cv2.circle(imgOriginal, (x, y), radius, (0, 0, 255), 3)
88              # end for
89          # end if
```

Abbildung 3-21 - Python Quellcode Kreise erkennen[69]

Zum Ende der Schleife bekommt die Datei Pos.dat einen Schreibschutz.

```
90          os.system("chmod 0400 Pos.dat")
91          # create windows, use WINDOW_AUTOSIZE for a fixed window size
92          cv2.namedWindow("imgOriginal", cv2.WINDOW_AUTOSIZE)
93          # or use WINDOW_NORMAL to allow window resizing
94          cv2.namedWindow("imgThresh", cv2.WINDOW_AUTOSIZE)
95          # show windows
96          cv2.imshow("imgOriginal", imgOriginal)
97          cv2.imshow("imgThresh", imgThresh)
98
99      # end while
100
101     #Alle Fenster schließen
102     cv2.destroyAllWindows()
103     return
```

Abbildung 3-22 - Python Quellcode Ende[70]

Um das Programm nun auch mal zu testen muss die Datei ins Homeverzeichnis vom Raspberry kopiert werden. Danach kann das Programm über das Terminal gestartet werden:

```
source ~/.profile
workon cv
python OpenCV_RedPoint_Detect.py
```

[69] Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py

[70] Quelle: Ausschnitt aus dem Programm OpenCV_RedPoint_Detect.py

4 Zusammenfassung und Ausblick

Der Umfang für die Grundlagenforschung ist deutlich größer ausgefallen als zuvor angenommen, weswegen das System nicht final getestet werden konnte. Die fehlenden Punkte müssen in einer weiteren Projektarbeit bearbeitet werden.

Dazu gehört:

- Programm Code Anpassung
  - Erkennung von Rechteckigen Objekten und Ermittlung der Koordinaten
  - Ermittlung von einer Rotation
  - Kontrolle der Größe der zu erkennenden Objekte
  - Schnittstelle zur Datenübertragung der Daten
- Bewertung
  - Allgemeiner Funktionstest
  - Performance
  - Ausfallsicherheit
  - Störanfälligkeit

Ein Problem ist jedoch schon erkennbar, der Belichtungsmodus des Kameramoduls lässt sich über die OpenCV Schnittstelle scheinbar nicht konfigurieren. Wodurch jedes Bild in der Schleife andere Belichtungswerte haben kann.

Ist ein stark reflektierendes Objekt im Sichtfeld so wird das Bild dunkler. Ohne Objekt wird das Bild wieder heller. Mit diesem Effekt kann sich keine exakte Erkennung realisieren.

Darstellungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Alexander Stohr. „Camera Serial Interface“. *Wikipedia*, 7. Februar 2018. https://en.wikipedia.org/w/index.php?title=Camera_Serial_Interface&oldid=824381676.

Andreas Potthoff. „Die Raspberry Pi Kamera Module – Inbetriebnahme und Verwendung | ElectroDrome“. Zugegriffen 7. Juni 2018. https://electrodrome.net/die-raspberry-pi-kamera-module-inbetriebnahme-und-verwendung/.

Bernd Klein. „Python-Tutorial: Dateien lesen und schreiben“. Zugegriffen 9. Juni 2018. https://www.python-kurs.eu/dateien.php.

Chris Dahms. *Contribute to Raspberry_Pi_2_and_OpenCV_3_Tutorial_Part_1 development by creating an account on GitHub*. Python, 2018. https://github.com/MicrocontrollersAndMore/Raspberry_Pi_2_and_OpenCV_3_Tutorial_Part_1.

CODESYS. „CODESYS Store - CODESYS Control for Raspberry Pi SL“. Zugegriffen 8. Juni 2018. https://store.codesys.com/systeme/codesys-control-for-raspberry-pi-sl.html#1.

———. „CODESYS Store - OpenCV for Raspberry Pi“. Zugegriffen 8. Juni 2018. https://store.codesys.com/opencv-for-raspberry-pi.html.

———. „CODESYS Store - Systeme“. Zugegriffen 8. Juni 2018. https://store.codesys.com/systeme.html?p=2.

———. „Das System“. Zugegriffen 8. Juni 2018. https://de.codesys.com//das-system.html.

———. „Lizenzierung“. Zugegriffen 8. Juni 2018. https://de.codesys.com//das-system/lizenzierung.html.

DougieLawson. „Raspberry Pi Forums“. Zugegriffen 8. Juni 2018. https://www.raspberrypi.org/forums/viewtopic.php?t=68247.

Ebay. „2x Arducam OV5647 CS Mount Kameramodul für Raspberry Pi mit 4 und 6mm Objektiv“. eBay. Zugegriffen 7. Juni 2018. https://www.ebay.de/itm/2x-Arducam-OV5647-CS-Mount-Kameramodul-fuer-Raspberry-Pi-mit-4-und-6mm-Objektiv-/263647321059.

Hilscher. „NetHAT“. netIOT, 16. August 2017. www.netiot.com.

K, Abid. *OpenCV2-Python-Tutorials: This repo contains tutorials on OpenCV-Python library using new cv2 interface*. Python, 2018. https://github.com/abidrahmank/OpenCV2-Python-Tutorials.

Kiwi Electronics. „PiFace GPIO Shim für Raspberry Pi“. Kiwi Electronics. Zugegriffen 7. Juni 2018. https://www.kiwi-electronics.nl/piface-gpio-shim.

Lang, Robert. „OpenCV for Raspberry Pi“, o. J. Zugegriffen 7. Juni 2018.

Lars Fermum. „Optische Grundlagen“. Zugegriffen 7. Juni 2018. http://www.vision-doctor.com/optische-grundlagen.html.

Michael Kofler. *Raspberry Pi das umfassende Handbuch*, 2017. http://nbn-resolving.de/urn:nbn:de:101:1-201708226939.

———. „WLAN schon vor der Inbetriebnahme konfigurieren | pi-buch.info“. Zugegriffen 8. Juni 2018. https://pi-buch.info/wlan-schon-vor-der-installation-konfigurieren/.

OpenCV. „OpenCV library“. Zugegriffen 8. Juni 2018. https://opencv.org/.

Patrick Schnabel. „Raspberry Pi 3 Modell B“. Zugegriffen 6. Juni 2018. https://www.elektronik-kompendium.de/sites/raspberry-pi/2102291.htm.

———. „Samba-Freigabe auf dem Raspberry Pi einrichten“. Zugegriffen 8. Juni 2018. https://www.elektronik-kompendium.de/sites/raspberry-pi/2007071.htm.

Peter J. Vis. „Raspberry Pi CSI-2 Connector Specifications“. Zugegriffen 7. Juni 2018. https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI-2_Connector_Specifications.html.

Pimoroni. „Raspberry Pi Camera HDMI Cable Extension — Pimoroni“. Zugegriffen 7. Juni 2018. https://shop.pimoroni.de/products/pi-camera-hdmi-cable-extension.

Prof. Dr.-Ing. habil. Karl-Heinz Meusel. „Digitaltechnik - FPGA“. WS 2016.

„Raspberry Pi Camera Module - Raspberry Pi Documentation“. Zugegriffen 8. Juni 2018. https://www.raspberrypi.org/documentation/raspbian/applications/camera.md.

Raspberry Pi Foundation. „Schematics Raspberry Pi“. Zugegriffen 7. Juni 2018. https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3b_1p2_reduced.pdf.

Raspberry Tips. „SSH am Raspberry Pi aktivieren – Neue Methode mit Raspbian 25.11.2016“. *raspberry.tips* (blog), 23. Dezember 2016. https://raspberry.tips/raspberrypi-tutorials/ssh-am-raspberry-pi-aktivieren-neue-methode-mit-raspbian-25-11-2016.

Rosebrock, Adrian. „Raspbian Stretch: Install OpenCV 3 + Python on Your Raspberry Pi“. *PyImageSearch* (blog), 4. September 2017. https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/.

Schmidt, Harald. *Raspberry Pi programmieren mit C/C++ und Bash: Mehr als 50 Programme rund um Foto, Video & Audio*. München: Hanser, 2018.

Seitz, Matthias. *Speicherprogrammierbare Steuerungen für die Fabrik- und Prozessautomation*. 4., überarbeitete und erweiterte Auflage. München: Fachbuchverlag Leipzig im Carl Hanser Verlag, 2015.

Steve Chamberlin. „Raspberry Pi GPIO Programming in C | Big Mess o’ Wires“. Zugegriffen 7. Juni 2018. https://www.bigmessowires.com/2018/05/26/raspberry-pi-gpio-programming-in-c/.

wiki.seeedstudio.com. „Raspberry Pi 3 Model B“. Zugegriffen 6. Juni 2018. http://wiki.seeedstudio.com/Raspberry_Pi_3_Model_B/.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Dies gilt auch für Quellen aus eigenen Arbeiten.

Ich versichere, dass ich diese Arbeit oder nicht zitierte Teile daraus vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Mir ist bekannt, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs mittels einer Plagiatserkennungssoftware auf ungekennzeichnete Übernahme von fremdem geistigem Eigentum überprüft werden kann.

Ich versichere, dass die elektronische Form meiner Arbeit mit der gedruckten Version identisch ist.

Datum, Unterschrift

Lebenslauf

Name:	Leo Enns
Geburtsdatum:	07.03.1990
Geburtsort:	Waldbröl (NRW)
Familienstand:	ledig
Wohnsitz:	Waldbröl (NRW)

Elementarausbildung:	1996 - 2009
Schulabschluss:	Allgemeine Hochschulreife (Gymnasium)

Ausbildung: 2009 – 2012

Ausbildung bei der Firma GIZEH Verpackungen GmbH & Co. KG in Bergneustadt mit dem Abschluss als Fachinformatiker – Systemintegration

Beruf: 2012 – 2017

Angestellter der Firma GIZEH Verpackungen GmbH & Co. KG in Bergneustadt, im Bereich:

- IT
- Elektrische Maschineninstandhaltung
- Technischer Einkauf
- Programmierung von Sondermaschinen und Roboter

Beruf: seit 2018

Angestellter der Firma DALEX Schweißmaschinen GmbH & Co. KG in Wissen, im Bereich:

- Programmierung von Sondermaschinen und Roboter

Studium: seit 01.09.2013

an der Rheinischen Fachhochschule, Köln Fachrichtung Elektrotechnik