

---

# DIPLOMARBEIT

## Implementierung eines Home-Security-Systems (HSS)

Höhere Technische Bundeslehr- und Versuchsanstalt Anichstraße

---

### Abteilung

Elektronik und technische Informatik

**Ausgeführt im Schuljahr 2020/21 von:**

Toprak Berke - 5BHEL

Stojanović Đorđe - 5BHEL

**Betreuer/Betreuerin:**

Ing. Signitzer Markus, Msc.

Innsbruck, am 26. März 2021

---

Abgabevermerk:

Datum:

Betreuer/in:

---

## **ERKLÄRUNG DER EIGENSTÄNDIGKEIT DER ARBEIT**

### **Eidesstattliche Erklärung**

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

---

Ort, Datum

Verfasser: Berke Toprak

---

Ort, Datum

Verfasser: Stojanović Đorđe

---

## **Gendererklärung**

Aus Gründen der besseren Lesbarkeit wird in dieser Diplomarbeit die Sprachform des generischen Maskulinums angewendet. Es wird an dieser Stelle darauf hingewiesen, dass die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig verstanden werden soll.

---

## **ZUSAMMENFASSUNG DES PROJEKTERGEBNISSES**

### **Kurzfassung**

Die vorliegende Diplomarbeit beschäftigt sich mit der Entwicklung und Produktion eines Home-Security-Systems, kurz HSS, welches eine Echtzeitüberwachung eines Bereichs, inklusive eines grafischen Monitorings, auf einer global-zugreifbaren, selbsterstellten Webseite, ermöglicht. Zur Überwachung werden Temperaturdaten, Luftfeuchtigkeitsdaten und lokale Videoaufnahmen (Liveübertragung) aufgezeichnet. Am Webserver können somit alle aufgezeichneten Daten beziehungsweise Informationen abgerufen werden. Die Daten des Bewegungsmelders werden in der Liveübertragung erkenntlich gemacht. Das HSS besteht aus einem Raspberry Pi als Hauptkomponente beziehungsweise „Motherboard“. Weitere Komponenten (der Temperatur- und Luftfeuchtigkeitssensor, die Kamera und der Bewegungssensor) werden zur Datenaufnahme ans Gehäuse verbaut und mit dem Motherboard verbunden. Das gesamte System ist in einem 3D-gedrucktem Gehäuse untergebracht. Die Grundidee ist eine kostengünstige Möglichkeit wichtige Informationen eines Bereichs aufzuzeichnen und diese live zu betrachten.

### **Abstract**

This diploma thesis deals with the development and production of a home security system, HSS for short, which enables real-time monitoring of an area, including graphical monitoring, on a globally accessible, self-created website. For monitoring, temperature data, humidity data, and local video recordings (livestream) are recorded. All recorded data and information can be accessed on the webserver. The data of the motion detector is made visible in the live-stream. The HSS consists of a Raspberry Pi as the main component or "motherboard". Other components (the temperature and humidity sensor, the camera, and the motion sensor) are mounted on the housing and connected to the motherboard to record data. The entire system is housed in a 3D-printed case. The basic idea is a cost-effective way to record important information about an area and view it live.

---

## **Projektergebnis**

Alle vorgegebenen Projektziele wurden erfüllt. Die Komponenten wurden hardwaremäßig in das Gehäuse verbaut und mit dem Raspberry Pi verbunden. Die Komponenten wurden softwaremäßig programmiert und konfiguriert. Es wurde eine Datenbank erstellt, in welches das Hauptprogramm die ausgelesenen Daten speichert. Anschließend wurden die Daten der Datenbank auf dem Webserver grafisch dargestellt. Die Kamera wird in diesem Fall als ein dynamisches Bauteil verwendet (Liveübertragung), deshalb lässt sich diese auf einer eigenen Subdomain einsehen. Diese Subdomain ist mit einem Button am Webserver erreichbar. Es wurde zudem ein Web-Zertifikat erstellt, durch welches man mittels Port-Forwarding, global auf den Webserver zugreifen kann. In dieser Arbeit wurde der 443 Port (HTTPS) verwendet. Als Zusatzfunktion war in Planung, den Webserver per dynamischer DNS (öffentliche URL) zugreifbar zu machen. Nach mehreren Problemen und dazugehörigen Meetings mit dem Projektbetreuer ist dies gescheitert und wurde ausgelassen. Der Webserver ist demnach nur per IP-Adresse aufrufbar (sobald man diese kennt). Dies stellt allerdings keinen Nachteil des Systems dar und bringt keine Beeinträchtigung mit sich. All das wurde in einem selbst-designten, schwarzen 3D-gedrucktem Gehäuse (Blackbox) eingebaut. Die dafür benötigten Aussparungen für die Komponenten (z. B.: Kamera) wurden schon beim Designen berücksichtigt. Die Gehäusebohrungen wurden in der Werkstatt der HTBLVA Anichstraße nachbearbeitet (z. B.: zur Befestigung der Komponenten).

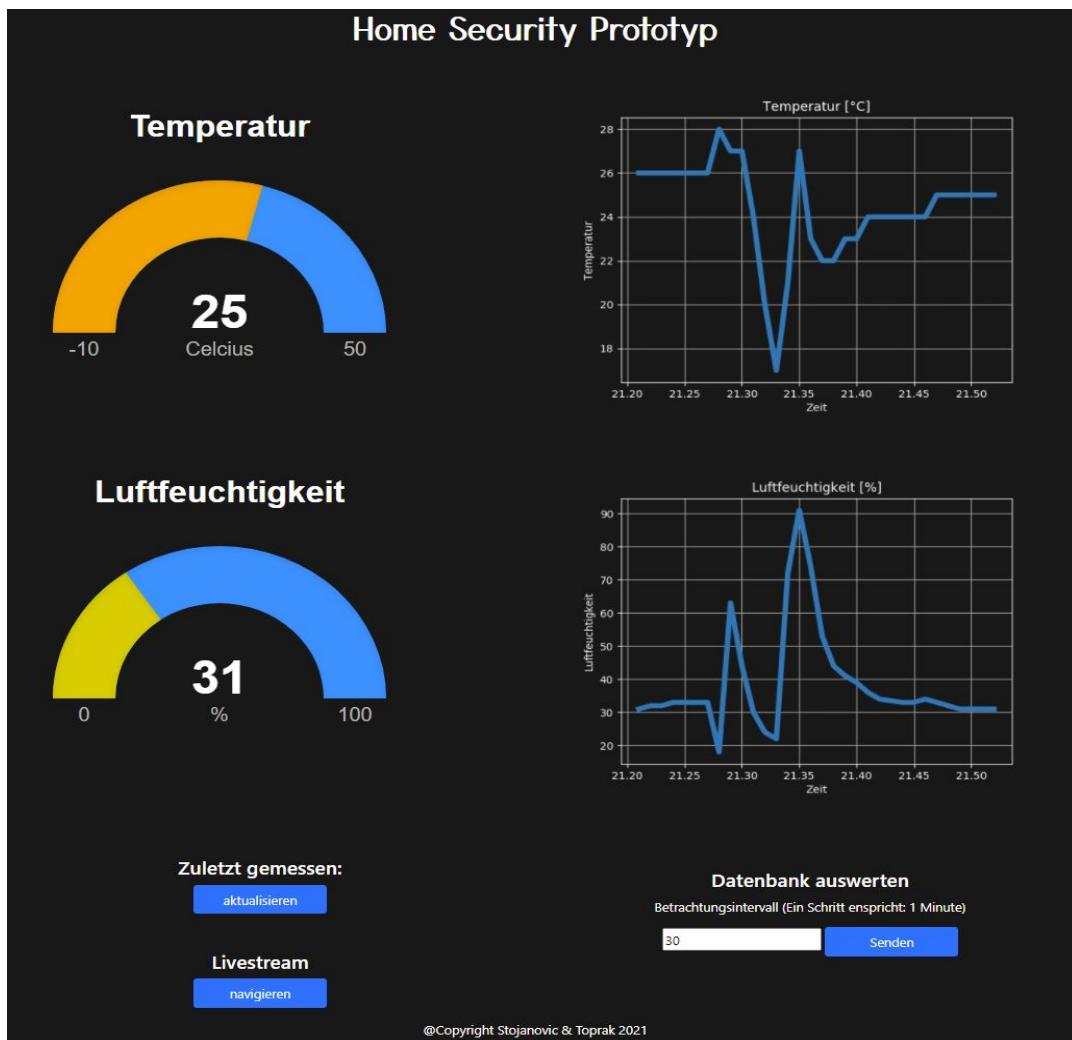


Abbildung 1: Webserver-Dashboard

Hier sieht man das Dashboard. Diese Seite ist nur mit der IP-Adresse aufrufbar. Es wurde sehr viel Wert auf Design und korrekte Farbkorrelation gelegt. Links im Bild sieht man in einem Halbringdiagramm den jetzigen Temperatur- und Luftfeuchtigkeitsstand. Dieser reagiert auf Veränderung und ändert sich, sobald man auf „aktualisieren“ (Button unten links) klickt. Die Farbe der Balken (im Bild derzeitig in orange/gelb abgebildet) ändert sich im direkten Verhältnis mit der Temperatur oder mit der Luftfeuchtigkeit. Beispiel: Je wärmer es im Raum ist, desto rötlicher wird der Balken. Rechts im Bild sieht man bezogen auf die historische Veränderung der Datenbankeinträge das dazugehörige XY-Liniendiagramm (Y-Achse = ausgegebene Daten, X-Achse = Zeit). Unter den Liniendiagrammen findet man ein Feld, wo man die gewünschte Anzahl der Datenbankeinträge eingeben kann. Nachdem man einen gewünschten Wert eingegeben hat (im Bild „30“) klickt man auf „senden“. Nun bekommt man die gewünschten Daten zu sehen - in diesem Fall die letzten 30.

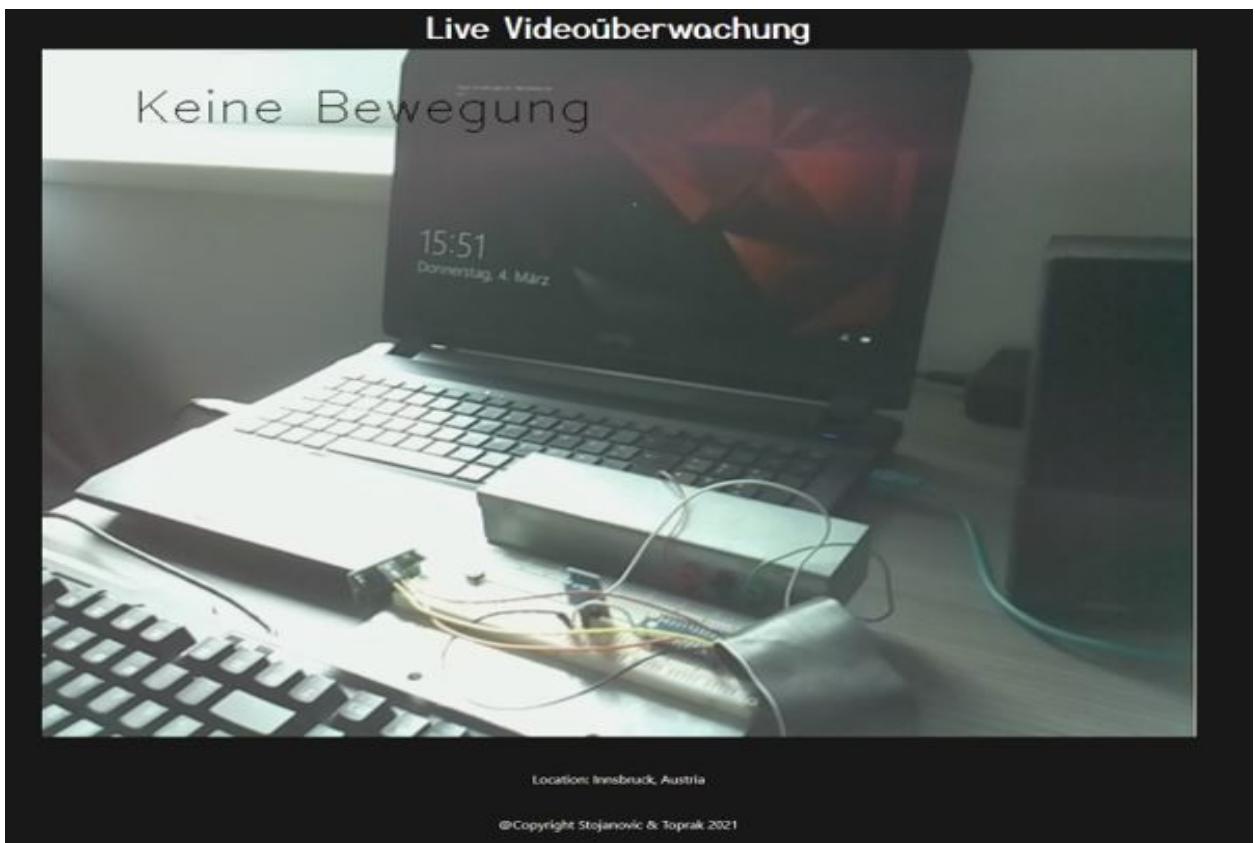


Abbildung 2: Webserver-Liveübertragung

Hier sieht man die Live-Videoüberwachung, den man über den Button „Livestream - navigieren“ erreichen kann. Diese Live-Videoüberwachung hat eine eigene Subdomain. Es werden durchschnittlich 30 Bilder pro Sekunde übertragen und die Qualität (runtergedrosselt) liegt bei 480p. Oben links sieht man die Daten des Motion-Sensors beziehungsweise die Detektion des Bewegungsmelders. Bei einer Bewegungsfeststellung wird der Text „keine Bewegung“ mit „Bewegung“ getauscht, nach einer Weile wird der Text wieder zurückgesetzt und der Vorgang wiederholt sich. Unter dem Livebild wird die derzeitige Lokation angezeigt.



Abbildung 3: Fertiges Produkt

Hier sieht man das fertige Produkt. Ein voll funktionsfähiges Überwachungssystem mit den nötigen Aufzeichnungsdaten (Bild, Temperatur, Luftfeuchtigkeit). Dieser Prototyp ist ein Grundgerüst eines einfachen, kostengünstigen Überwachungssystems, das einfach zum Aufrüsten ist. Das war eines der Ziele. Daher ist dies nur mit wenigen Features ausgestattet - siehe Punkt Alternative Lösungen (7.) um Ideen und Gedanken für Zusatzfunktionen zu erfahren. Im Bild sieht man die Aussparung für den LAN- beziehungsweise USB-Port. Gut sichtbar sind auch die integrierten Komponenten. Dabei ist beachtet, dass die Kamera nicht aus dem Gehäuse herausragt, um bei Stürzen die Linse zu schützen, da diese aus Glas besteht und leicht brechen oder zerkratzen könnte. Der Motion-Sensor (in weiß) muss jedoch aus dem Gehäuse ragen, um wirklich möglichst alles in seinem Radius zu erkennen, um es anschließend in der Liveübertragung angezeigt werden kann. Der Temperatur- und Luftfeuchtigkeitssensor misst dabei die Umgebungstemperatur und gibt diese auf dem Webserver grafisch an.

---

## **Danksagung**

An dieser Stelle möchten wir, Stojanović Đorđe und Toprak Berke, uns bei unserem Lehrer und Projektbetreuer Herr Ing. Signitzer Markus, MSc, herzlichst bedanken, der uns während der Entwicklung und Anfertigung dieser Diplomarbeit sehr unterstützt und motiviert hat.

Zudem möchten wir uns bei unseren Jahrgangsvorständen, Herr DI. Mag. Walter Erich, M.A. und Herr Mag. Eller Johannes, bedanken, welche stets ein offenes Ohr für uns hatten und Klasse Arbeit geleistet haben.

Abschließend möchten wir uns herzlichst bei unseren Eltern  
(Dejan Stojanović und Jasna Stojanović bzw. Yilmaz Toprak und Selma Toprak)  
für die jahrelange Begleitung während unserer gesamten HTL-Karriere und die Unterstützung während der Diplomarbeit bedanken.

Besonderen Dank an Yilmaz Toprak (INNIO Jenbacher GmbH) für die spontane Unterstützung bei der Produktion des 3D-gedruckten Gehäuses.

# Inhaltsverzeichnis

<b>Erklärung der Eigenständigkeit der Arbeit</b> . . . . .	i
<b>Gendererklärung</b> . . . . .	ii
<b>Kurzfassung</b> . . . . .	iii
<b>Projektergebnis</b> . . . . .	iv
<b>Danksagung</b> . . . . .	viii
<b>1 EINLEITUNG</b> . . . . .	1
1.1 Projektbeschreibung . . . . .	1
1.2 Sinn und Zweck . . . . .	1
<b>2 VERTIEFENDE AUFGABENSTELLUNG</b> . . . . .	2
2.1 Toprak Berke . . . . .	2
2.2 Stojanović Đorđe . . . . .	2
<b>3 THEORETISCHE GRUNDLAGEN UND VORWISSEN</b> . . . . .	3
3.1 Grundlagen: Hardware . . . . .	3
3.1.1 Raspberry Pi . . . . .	3
3.1.2 Raspberry Pi Cam Rev 1.3 . . . . .	5
3.1.3 Bewegungsmelder . . . . .	7
3.1.4 Temperatur- und Luftfeuchtigkeitssensor . . . . .	8
3.2 Grundlagen: Software . . . . .	9
3.2.1 Programmiersprachen . . . . .	9
3.2.2 Interpreter und Compiler . . . . .	9
3.2.3 Objektorientierung . . . . .	10
3.2.4 Python . . . . .	10
3.2.5 WSGI . . . . .	10
3.2.6 Flask . . . . .	10
3.2.7 Auszeichnungssprachen und Programmiersprachen . . . . .	11
3.2.8 HTML, CSS und JS . . . . .	12
3.2.9 Datenbanken . . . . .	13
3.2.10 SQLite . . . . .	13
3.3 Grundlagen: Netzwerk . . . . .	14

3.3.1	7-Schichten-OSI-Modell . . . . .	14
3.3.2	IPv4-Adressen . . . . .	16
3.3.3	Portweiterleitung . . . . .	16
3.3.4	SSL-Zertifikate . . . . .	16
3.4	Grundlagen: 3D-Druck . . . . .	17
3.4.1	Definition des 3D-Drucks . . . . .	17
3.4.2	Materiakunde . . . . .	17
3.4.2.1	Materialien . . . . .	17
3.4.2.2	Verwendung im Projekt . . . . .	18
<b>4</b>	<b>WAHL DER ARBEITSMEDIEN . . . . .</b>	<b>19</b>
4.1	Hardwareauswahl . . . . .	19
4.1.1	Raspberry 3b Plus . . . . .	19
4.1.2	Zusätzliche Komponenten . . . . .	20
4.1.2.1	Kamera . . . . .	20
4.1.2.2	Motion-Sensor . . . . .	21
4.1.2.3	Temperatur- und Luftfeuchtigkeitssensor . . . . .	22
4.2	Softwareauswahl . . . . .	23
4.2.1	Programmiersprachen und Frameworks . . . . .	23
4.2.1.1	Python . . . . .	23
4.2.1.2	Flask . . . . .	23
4.2.1.3	SQLite . . . . .	23
4.2.1.4	HTML, CSS und JS . . . . .	24
4.2.2	Weitere Software, Plugins, Frameworks und Libraries . . . . .	24
4.2.2.1	Adafruit DHT Sensor Library - <i>Python Library</i> . . . . .	24
4.2.2.2	Matplotlib - <i>Python Library</i> . . . . .	24
4.2.2.3	jQuery - <i>JavaScript Library</i> . . . . .	24
4.2.2.4	Raphaël - <i>JavaScript Library</i> . . . . .	24
4.2.2.5	JustGage - <i>JavaScript Plugin</i> . . . . .	24
4.2.2.6	Bootstrap - <i>CSS Framework</i> . . . . .	24
4.2.2.7	OpenCV - <i>freie Library</i> . . . . .	24
4.3	Basis . . . . .	25
4.3.1	Erklärung . . . . .	25
4.3.2	Verwendungszwecke und Unterschiede . . . . .	26
<b>5</b>	<b>PRAKТИSCHE UMSETZUNG . . . . .</b>	<b>27</b>

5.1	Vorbereitung des Raspberry Pi 3B Plus . . . . .	27
5.1.1	Flashen der SD-Karte mit Raspberry Pi OS . . . . .	27
5.1.2	Erstellung des Programmverzeichnisses . . . . .	33
5.2	Software- und Hardwaretechnische Umsetzung der Bauteile . . . . .	34
5.2.1	PIR Motion Sensor . . . . .	34
5.2.1.1	Grundlegende Funktionsbeschreibung . . . . .	34
5.2.1.2	Hardwaretechnische Umsetzung . . . . .	34
5.2.1.3	Überblick aller Programmfunctionen . . . . .	34
5.2.1.4	Softwaretechnische Implementierung . . . . .	35
5.2.2	Kamera . . . . .	36
5.2.2.1	Grundlegende Funktionsbeschreibung . . . . .	36
5.2.2.2	Hardwaretechnische Umsetzung . . . . .	36
5.2.2.3	Installation der Kamera . . . . .	36
5.2.2.4	Installation der OpenCV Library . . . . .	37
5.2.2.5	Das Kameraproblem und die Lösung: . . . . .	38
5.2.3	Datenbank . . . . .	39
5.2.3.1	Grundlegende Funktionsbeschreibung . . . . .	39
5.2.3.2	Installation von SQLite . . . . .	39
5.2.3.3	Softwaretechnische Ausarbeitung der Datenbank . . . . .	39
5.2.4	DHT 11 . . . . .	41
5.2.4.1	Grundlegende Funktionsbeschreibung . . . . .	41
5.2.4.2	Hardwaretechnische Umsetzung . . . . .	41
5.2.4.3	Installation der ADAFRUIT Library . . . . .	41
5.2.4.4	DHT 11 Test . . . . .	42
5.2.4.5	Überblick aller Programmfunctionen . . . . .	43
5.2.4.6	Softwaretechnische Ausarbeitung aller Programmfunctionen . . . . .	43
5.3	Hauptprogramm . . . . .	46
5.3.1	Aufsetzen der Ordnerstruktur . . . . .	46
5.3.2	Backend . . . . .	47
5.3.2.1	Grundlegende Funktionsbeschreibung . . . . .	47
5.3.2.2	Überblick aller Programmfunctionen . . . . .	47
5.3.2.3	Installation von Matplotlib . . . . .	48
5.3.2.4	Installation von Flask . . . . .	48
5.3.2.5	Softwaretechnische Implementierung . . . . .	49
5.3.2.6	Vorbereitung des Fernzugriffs . . . . .	61
5.3.3	Frontend . . . . .	62

5.3.3.1	Installation JustGage . . . . .	62
5.3.3.2	Finale Ordnerstruktur . . . . .	62
5.3.3.3	[ index.html ] . . . . .	63
5.3.3.4	[ kamera.html ] . . . . .	67
5.3.3.5	[ style.css ] . . . . .	69
5.4	Fernzugriff . . . . .	70
5.4.1	Programmintern . . . . .	70
5.4.2	Programmextern . . . . .	70
5.4.3	Zertifikatsergebnis . . . . .	71
5.5	Gehäuse: Entwicklung, Produktion und Nachbearbeitung . . . . .	72
5.5.1	Selbstgestellte Bedingungen bezogen auf Funktionalität . . . . .	72
5.5.2	Informationen zum Druck . . . . .	73
5.5.3	Skizzen und Ideen . . . . .	75
5.5.3.1	Engere Auswahl . . . . .	75
5.5.3.2	Entgültige Auswahl . . . . .	77
5.5.4	Raspberry Pi 3B Plus Dimensionen . . . . .	78
5.5.5	Entwicklung und Design der Blackbox . . . . .	79
5.5.5.1	Design in AutoCad . . . . .	79
5.5.5.1.1	AutoCad: Unterseite . . . . .	80
5.5.5.1.2	AutoCad: Deckel . . . . .	81
5.5.5.1.3	AutoCad: Seiten . . . . .	82
5.5.5.1.4	AutoCad: Vorderseite . . . . .	83
5.5.5.1.5	AutoCad: Rückseite . . . . .	84
5.5.5.2	Design in Catia . . . . .	85
5.5.5.2.1	Catia: Deckel . . . . .	85
5.5.5.2.2	Catia: Basis . . . . .	86
5.5.6	3D-Druck . . . . .	87
5.5.7	Nachbearbeitung des Gehäuses . . . . .	88
5.5.7.1	Bohrungen . . . . .	88
5.5.7.2	Power-Port . . . . .	89
5.6	Projektschematik . . . . .	90
<b>6</b>	<b>PERFORMANCETESTS</b> . . . . .	<b>91</b>
<b>7</b>	<b>ALTERNATIVE LÖSUNGSWEGE</b> . . . . .	<b>93</b>
<b>8</b>	<b>SCHLUSSFOLGERUNG UND PROJEKTERFAHRUNG</b> . . . . .	<b>95</b>

<b>9 PROJEKTTERMINPLANUNG . . . . .</b>	<b>96</b>
<b>10 MEILENSTEINE . . . . .</b>	<b>97</b>
10.1 Toprak Berke . . . . .	97
10.2 Stojanović Đorđe . . . . .	97
<b>11 ARBEITSNACHWEIS . . . . .</b>	<b>98</b>
11.1 Toprak Berke (179h) . . . . .	98
11.2 Stojanović Đorđe (176h) . . . . .	99
<b>12 KOSTENTABELLE . . . . .</b>	<b>100</b>
<b>Abbildungsverzeichnis . . . . .</b>	<b>I</b>
<b>Tabellenverzeichnis . . . . .</b>	<b>III</b>
<b>Codeverzeichnis . . . . .</b>	<b>IV</b>
<b>Literaturverzeichnis . . . . .</b>	<b>V</b>

---

# **1 EINLEITUNG**

## **1.1 Projektbeschreibung**

Im Rahmen dieser Diplom- beziehungsweise Projektarbeit soll ein Prototyp zur Überwachung eines Raumes entwickelt und produziert werden. Die handelsüblichen Überwachungssysteme sind sehr kostenintensiv und kompliziert. Dies hier ist eine kostengünstigere und unkomplizierte DIY-Variante eines aufstockbaren Überwachungssystems und somit ist der Nachbau auch für Nicht-Techniker geeignet.

## **1.2 Sinn und Zweck**

Dieser Prototyp zeichnet hauptsächlich Temperatur- und Luftfeuchtigkeitsdaten auf. Darüber hinaus kann man über die Live-Videoübertragung (mit integriertem Bewegungsmelder) das Geschehen beobachten. Somit ist dieses Home-Security-System für Überwachungen von zum Beispiel Gewächshäusern, wo genau eine Temperatur- und eine Luftfeuchtigkeitsmessung vollauf nötig ist, ideal. Auch die Überwachung von kleinen Serverräumen ist für dieses Gerät kein Problem. Ein riesiger Vorteil ist es, dass man sich nicht im gleichen Netzwerk befinden muss, um auf den Webserver zuzugreifen. Der Webserver ist global erreichbar. Das HSS erleichtert viele alltägliche Situationen und das sehr kostengünstig und relativ unkompliziert. Fragen bezüglich der Temperaturveränderung in der letzten Nacht oder ob jemand gerade im Raum ist, werden wortwörtlich mit einem Klick beantwortet.

---

## **2 VERTIEFENDE AUFGABENSTELLUNG**

### **2.1 Toprak Berke**

Toprak Berke ist für die Programmierung aller Teilkomponenten zuständig. Darüber hinaus ist er noch verantwortlich für die Implementierung und die Konfiguration des Raspberry-Pi-Security-Systems. Die Datenmigration beziehungsweise die korrekte Übergabe dieser einzelnen Daten an den Webserver ist zu garantieren. Er gewährleistet und konfiguriert den Fernzugang des Webservers am Router. Außerdem ist er für die Entwicklung und Produktion einer Blackbox beziehungsweise des Gehäuses zuständig.

### **2.2 Stojanović Đorđe**

Stojanović Đorđe ist der Verantwortliche für die Programmierung des eigentlichen Webserver mittels Python mit dem Webframework Flask. Er ist der Zuständige für eine Datenbank, welche, in einem Minutentakt, Daten speichern soll. Außerdem ist er zuständig für das grafische Monitoring auf dem Webserver - aufgebaut aus den erfassten beziehungsweise zuvor migrierten Daten der Datenbank. Darüber hinaus ist er verantwortlich für die Entwicklung der Fernsteuerung am Raspberry PI auf den Webserver.

---

### 3 THEORETISCHE GRUNDLAGEN UND VORWISSEN

#### 3.1 Grundlagen: Hardware

##### 3.1.1 Raspberry Pi

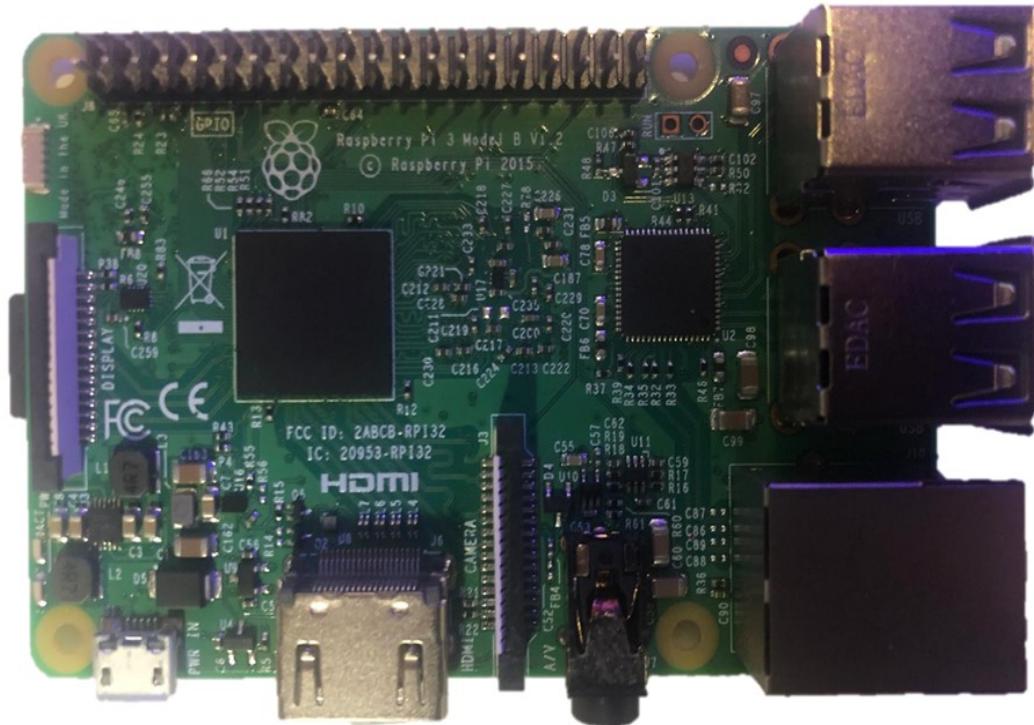


Abbildung 4: Raspberry Pi 3B Plus Platine

Sprich *raspberry pie*. Ein Himbeer-Kuchen als Computer? Manche Tech-Hersteller benennen die Produkte, die Sie herstellen, nach Früchten - zum Beispiel „Apple“. Doch das „Pi“ im Wort Raspberry Pi hat nicht die Bedeutung eines Kuchens (Pie -> nicht Pi). Die Bedeutung von Pi ist: „Python interpreter“. Python ist eine vorinstallierte Programmiersprache und wird als die Standardprogrammiersprache des Raspberry Pi gesehen. Näheres über Python und andere Programmiersprachen wird in den Punkten ab 3.2 näher erklärt. Der Raspberry Pi ist im Gegensatz zum Arduino ein voll funktionsfähiger Computer mit entsprechender Grafikausgabe, Betriebssystem etc. Der RPi wurde 2012 auf den Markt gebracht, wobei der Erfolg an sich viel später kam. Erst als die breite Masse diesen Einplatinencomputer zum Programmieren lernen beziehungsweise zum Experimentieren benutzt hat, kam der Hype um die kreditkarten-große Platine. Das Ziel der Markteinführung war es, jungen, technikinteressierten Menschen das Programmieren leichter zu gestalten - genau aus diesem Grund wurden auch die Stück-

---

kosten sehr niedrig gehalten. Der eigentliche Grund für die Einführung waren die sinkenden Programmierfähigkeiten der jungen Menschen, die jedes Jahr an der Universität Cambridge statistisch belegt wurden. Nun wollte man die jungen Studenten nicht zusätzlich mit trockener Softwaretheorie plagen, sondern setzte ab 2012 auch auf den Raspberry Pi, um in das Studiengeschehen auch eine praktische Anwendung zu implementieren. Der RPi stellt standardmäßig frei programmierbare Input- und Outputpins zur Verfügung - in der Abbildung 4 ganz oben links. Diese werden auch GPIOs - sprich „General Purpose Input Output“- genannt. Dies stellt sozusagen die Kommunikation zwischen dem User und dem Raspberry Pi dar. Durch diese Schnittstellen kann der User Komponenten anschließen und diese frei nach eigenem Willen programmieren. In diesem Projekt werden auch die bereitgestellten GPIOs benutzt. Es können verschiedenste Bauteile angesprochen werden. So ist es sehr unkompliziert z.B. eine einfache Blinklichtsequenz mit einer LED durchzuführen, sobald ein Bewegungsmelder eine Aktion wahrnimmt. In diesem Fall müssen die zwei Komponenten an die Spannungsversorgung angehängt werden und noch dazu der Bewegungsmelder zu einem freien GPO für den Datenoutput. Zur direkten Videoausgabe, mit einer dazugehörigen Pi-Kamera, ist eine besondere Schnittstelle direkt auf dem Raspberry Pi intergriert. Die sogenannte CSI-Schnittstelle - sprich „Camera Serial Interface“. Diese liegt in der Abbildung 4 vertikal und direkt rechts neben dem HDMI-Port. Die Kamera, sowie alle anderen im Projekt benutzten Komponenten, werden in den folgenden Punkten genauer erklärt.<sup>1</sup>

---

<sup>1</sup>Quelle: Vgl. - 17.03.2021, <https://www.elektronik-kompendium.de/sites/com/1904221.htm>

### 3.1.2 Raspberry Pi Cam Rev 1.3

Auszug aus dem Datenblatt<sup>2</sup>

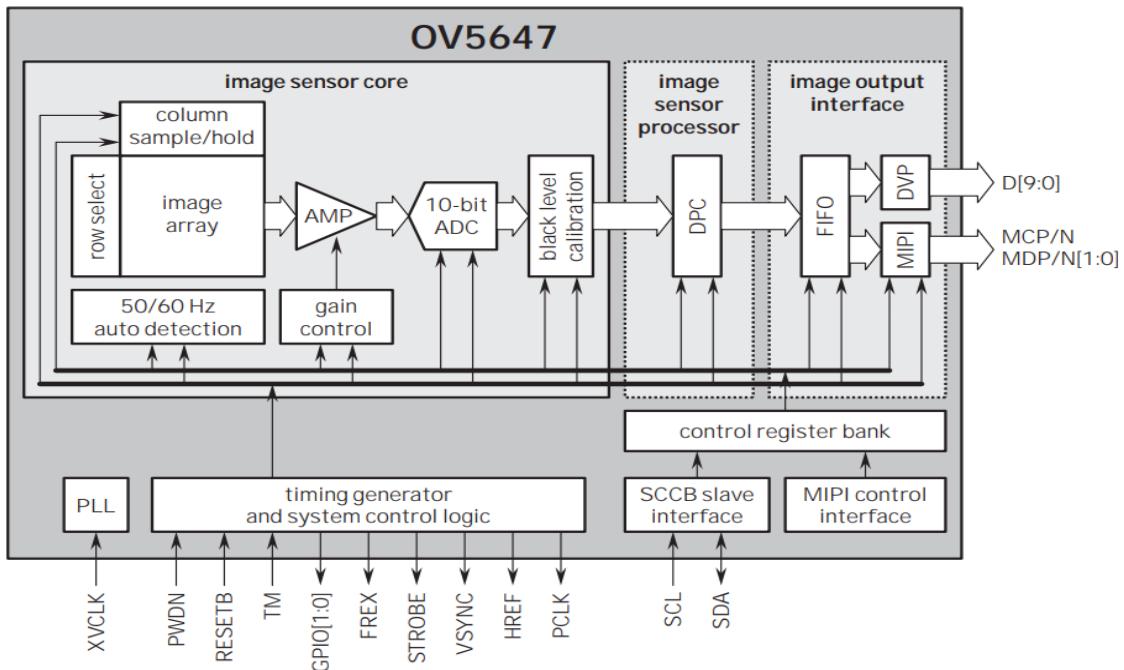


Abbildung 5: OmniVision 5647 Blockdiagramm

Dieses spezielle Raspberry Pi Kamera Modul ist mit einem Omnidision 5647 CMOS Sensor von OmniVision Technologies (Abbildung 5) ausgestattet. Dieser Sensor hat 5 Megapixel und das optische Format beziehungsweise die Linsengröße ist nur 6.25 mm. Wegen dieser Kompatibilität kann diese Kamera als Überwachungskamera genutzt werden, da sie, gerade wegen der kleinen Linse, schwer erkennbar ist und darüber hinaus keinen Performanceverlust bezogen auf die Qualität des Bildes mit sich zieht. Es ist noch hinzuzufügen, dass die Kamera ein Fixfokusobjektiv besitzt - mehr dazu bei Hardwareauswahl 4.1.2.1. Die Übertragungsqualität eines Bildes vom Sensor zum Raspberry Pi liegt bei 30 Fps bei 1080p, 45 Fps bei 960p und 60 Fps bei 720p. Da die Übertragungsqualität eines Live-Bildes beziehungsweise die Liveübertragung viel mehr an Leistung verbraucht und da dort auch erstmalig die Internetverbindung auch eine Rolle spielt, wurde die Qualität in der Live-Übertragung auf 480p runtergedrosselt, damit das Bild noch erkennbar flüssig bleibt. Das alles wird mit einem 15-bit Flachbandkabel an den Raspberry Pi gesendet - wobei 5 der 15 Pins nicht benutzt werden, da der OmniVision 5647 eine 10-bit Übertragung (10-bit ADC) in ein RAW RGB Format durchführt. Mehr Details bei der Hardwareauswahl des Kameras (4.1.2.1).<sup>3</sup>

<sup>2</sup>Quelle: 17.03.2021, [https://www.imagequalitylabs.com/PDFs/OmniVision\\_OV5647.pdf](https://www.imagequalitylabs.com/PDFs/OmniVision_OV5647.pdf)

<sup>3</sup>Quelle: Vgl. - 28.02.2021, <https://docs.rs-online.com/2888/0900766b8127db0a.pdf>

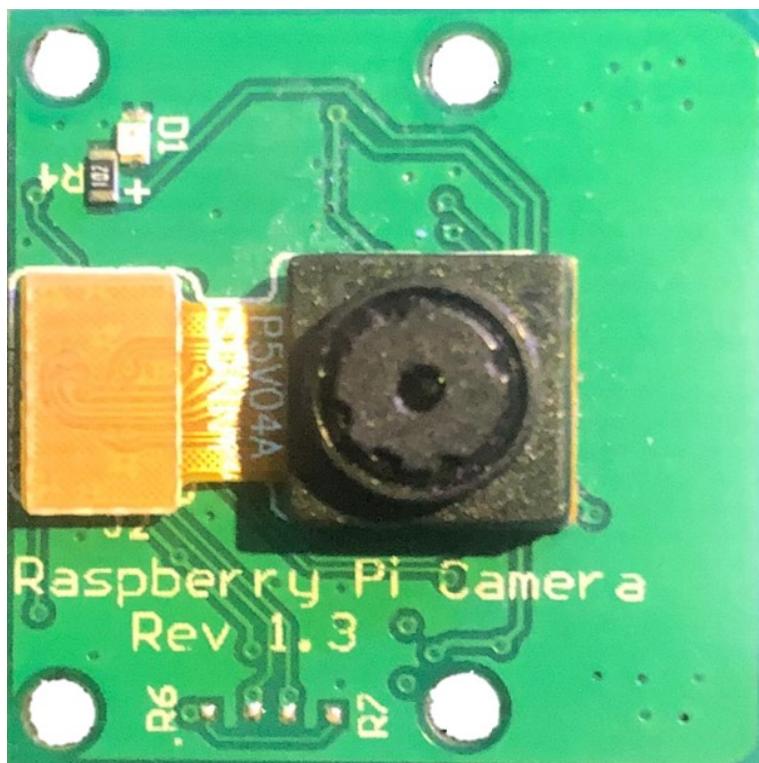


Abbildung 6: Raspberry Pi Cam Rev 1.3 Front

Die Raspberry Pi Cam Rev 1.3 wird mit vier 2 mm Schrauben am Gehäuse befestigt. Man muss dabei auch auf die richtige Platzierung des Winkels achten, da sonst das Bild verkehrt (z. B. auf dem Kopf) wäre. Dies müsste dann softwaremäßig korrigiert werden, um das Bild wieder richtig darzustellen. In diesem Projekt findet man das Beispiel einer Winkelkorrektur im Punkt 5.2.2.5.

---

### 3.1.3 Bewegungsmelder

Der HC-SR501 (siehe auch 4.1.2.2) Motion Sensor beziehungsweise Bewegungsmelder arbeitet mit der Infrarottechnologie und verwendet die LHI778 Sonde. Genauer gesagt ist dieser ein PIR-Sensor. PIR bedeutet „Pyroelektrischer Infrarot Sensor“. Unter Pyroelektrizität versteht man die Eigenschaft von piezoelektrischen Steinen (genauer gesagt Kristallen) mit einer Spannungsfreisetzung auf eine gewisse Temperaturänderung ( $\Delta T$ ) zu reagieren. Die Einsatzbereiche des HC-SR501, mit eingebauter Empfindlichkeits- und eingebauter Zeitverzögerungseinstellung, sind automatische Bewegungs- beziehungsweise Lichterkennungen für Lagerräume, Keller, Garagen etc. Auch einfache Alarmanlagen können mit diesem kleinen Bauteil realisiert werden. Der Motion Sensor sieht alles unter 110 beziehungsweise 120 Grad (Eventhorizont) und die horizontale Sichtweite beträgt ungefähr 7 Meter.<sup>4</sup> & <sup>5</sup>

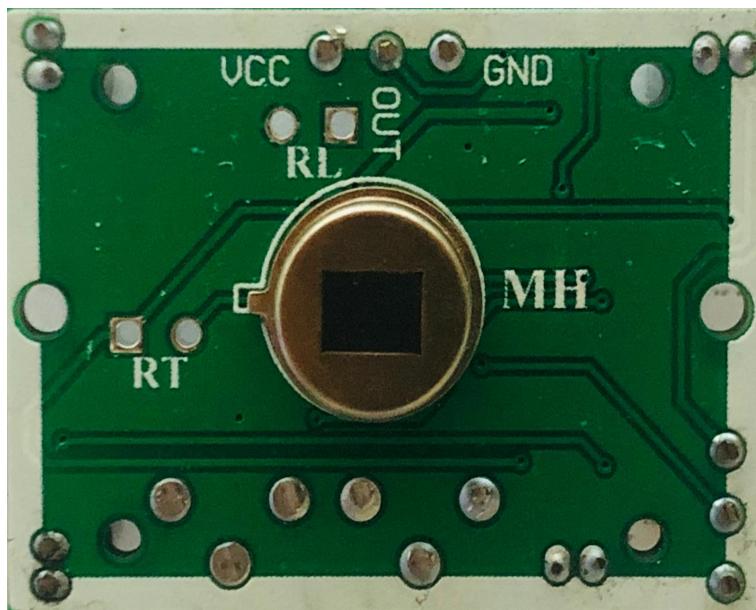


Abbildung 7: Motion Sensor ohne Abdeckung

---

<sup>4</sup>Quelle: Vgl. - 28.02.2021, <https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html>

<sup>5</sup>Quelle: Vgl. - 20.03.2021, <https://html.alldatasheet.com/html-pdf/14901/PERKINELMER/LHI778/605/1/LHI778.html>

---

### 3.1.4 Temperatur- und Luftfeuchtigkeitssensor

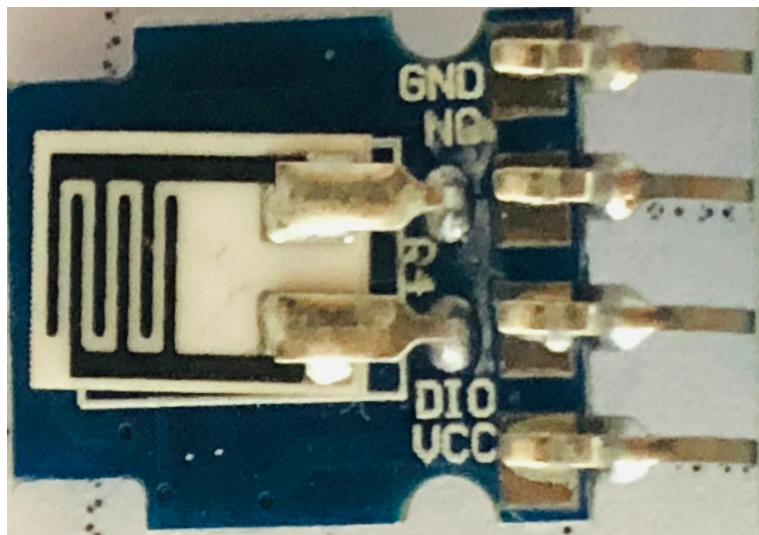


Abbildung 8: DHT 11 ohne Abdeckung

Der DHT11 Sensor (auch bekannt unter Temperatur- und Feuchtigkeitsmodul) arbeitet als Main-Komponente und ist daher eigentlich die wichtigste Komponente in diesem System, da es auch auf dem Webserver die Hauptdomain füllt (Subdomain = Liveübertragung). Das chinesische Datenblatt wurde für den Blackboxbau benutzt - hier wurden die Maße für den Ausschnitt entnommen. Die restlichen Daten und Informationen bezüglich Einsatzbereich, technische Bedingungen, Toleranzen, Spannungsversorgung etc. wurden aus einer technischen Webseite entnommen.<sup>6</sup>

Der DHT11 Sensor (auch der DHT22 Sensor) hat prinzipiell vier Pins. Pin 1 für die VCC Spannungsversorgung (3 bis 5V), der Pin 2 für den Datenoutput und Pin 4 für die Groundverbindung. Dabei ist Pin 3 nicht verbunden und spielt aus diesem Grund keine Rolle. Der DHT Sensor ist extrem anfällig und sehr empfindlich gegen zu hohe Ströme. Aus diesen Gründen muss ein Vorwiderstand (5 bis 10kOhm geschaltet werden. Der erfasste Temperaturbereich beträgt 0 bis 50 Grad Celsius und ist daher gut geeignet für alltägliche Situationen. Der erfasste Luftfeuchtigkeitsbereich liegt bei 20 bis 80% wobei wir diesen Bereich zur besseren Verständlichkeit grafisch mit 0 bis 100% darstellen. Das bedeutet es werden keine legitimen Werte unter 20% beziehungsweise über 80% gemessen. Gleicher gilt bei der Temperaturmessung. Der bessere DHT22 Sensor wird beim Punkt Alternative Lösungswege besser beschrieben (Punkt 7).<sup>7</sup>

---

<sup>6</sup>Quelle: Vgl. - 21.03.2021, <https://honey-pi.de/dht22-bzw-dht11-verwenden/>

<sup>7</sup>Quelle: Vgl. - 20.03.2021, <https://www.alldatasheet.com/datasheet-pdf/pdf/1132088/ETC2/DHT11.html>

---

## 3.2 Grundlagen: Software

### 3.2.1 Programmiersprachen

Eine Programmiersprache ist ausgelegt zum Schreiben von Computeranweisungen. Mit einer Programmiersprache kann die gewünschte Datenverarbeitung des Computers auf symbolische Art und Weise ausgedrückt werden. Dabei muss keine Rücksicht auf maschinenspezifische Details genommen werden.

### 3.2.2 Interpreter und Compiler

Beim Ausführen von Programmen muss der Computer den Programmcode auf eine gewisse Art und Weise interpretieren. Programmiersprachen haben dafür entweder einen Interpreter oder einen Compiler. Dies sind Programme, die den, von Programmierer erstellten Code, in Maschinencode umwandeln. Maschinencode ist meistens ein Binärcode (bestehend aus 1 und 0) welchen der Computer "versteht".

#### Unterschiede

Die Unterschiede der beiden Programme, werden in Tabelle 1 erläutert.<sup>8</sup>

Compiler	Interpreter
Der Compiler scannt das gesamte Programm in einem Durchgang.	Der Interpreter übersetzt das Programm Zeile für Zeile.
Am Scannende werden alle Fehler angezeigt.	Fehler werden Zeile für Zeile ausgegeben.
Vorteil - schnelle Ausführungszeit.	Nachteil - langsame Ausführungszeit.
Konvertiert Quellcode in Objektcode.	Scannt code Zeile für Zeile.
Nach dem Compilerprozess wird zur Ausführung kein Quellcode mehr benötigt.	Benötigt auch nach Interpreterprozess Quellcode zur Programmausführung.
Beispielprogrammiersprachen: C, C++, C#, ...	Beispielprogrammiersprachen: Python, Ruby, Perl, SNOBOL, MATLAB, ...

Tabelle 1: Unterschied: Compiler - Interpreter

---

<sup>8</sup>Quelle: 21.03.2021, <https://www.geeksforgeeks.org/difference-between-compiler-and-interpreter/>

---

### **3.2.3 Objektorientierung**

"Die objektorientierte Programmierung (OOP) verwendet "Objekte", um Objekte der realen Welt zu modellieren. Die objektorientierte Programmierung (OOP) besteht aus einigen wichtigen Konzepten, nämlich Kapselung, Polymorphismus, Vererbung und Abstraktion. Diese Merkmale werden allgemein als OOPS-Konzepte bezeichnet."<sup>9</sup>

### **3.2.4 Python**

Python ist eine Programmiersprache welche einen Interpreter verwendet. Die Sprache selbst ist objektorientiert (siehe 3.2.3.). Aufgrund der Abstraktheit wird Python öfters als "high-level programming language" bezeichnet. Der größte Vorteil von Python ist die leicht zu erlernende Programmsyntax, welche die Lesbarkeit erhöht und somit die Programmpflege erleichtert. Die Programmiersprache unterstützt Module und Pakete. Dies erleichtert zudem die Wiederverwendung von Code und verbessert die Modularität. Da Python Open-Source (frei einsehbar und kostenlos nutzbar) ist, sind der Python-Interpreter und die Standardbibliothek auf allen relevanten Plattformen kostenlos verfügbar.<sup>10</sup>

### **3.2.5 WSGI**

WSGI ist das "Web Server Gateway Interface". Durch das WSGI wird beschrieben wie Webserver mit Webanwendungen kommunizieren können. Außerdem beschreibt das WSGI wie Webanwendungen miteinander verkettet werden können, um Anfragen zu verarbeiten. WSGI ist ein Bestandteil von Python und elementar für die Umsetzung von Flask-Webanwendungen (Flask - siehe 3.2.6.).<sup>11</sup>

### **3.2.6 Flask**

Flask ist ein leichtgewichtiges WSGI-Webanwendungs-Framework. Flask wurde entwickelt, um einen sehr einfachen Einstieg in die Entwicklung von Webanwendungen zu garantieren. Dabei behält es jedoch die Fähigkeit auch für komplexeren Anwendungen skalierbar zu bleiben. Flask hat sich im Laufe der Jahre zu einem der beliebtesten Python-Webanwendungs-Frameworks entwickelt. Flask erzwingt kein festes Layout, sondern ist frei modulierbar. Eine Aufgabe kann in Flask auf viele, unterschiedliche Arten gelöst werden. Der Entwickler kann frei entscheiden, welche Werkzeuge und Programmbibliotheken er verwenden will.<sup>12</sup>

---

<sup>9</sup>Quelle: 22.03.2021, <https://javajee.com/summary-of-object-oriented-programming-oop-concepts-in-java>

<sup>10</sup>Quelle: Vgl. - 22.03.2021, <https://www.python.org/doc/essays/blurb/>

<sup>11</sup>Quelle: Vgl. - 17.01.2021, <https://wsgi.readthedocs.io/en/latest/what.html>

<sup>12</sup>Quelle: Vgl. - 17.01.2021, <https://palletsprojects.com/p/flask/>

---

### **3.2.7 Auszeichnungssprachen und Programmiersprachen**

Eine Auszeichnungssprache ist eine Computersprache, welche dazu verwendet wird, um Strukturen zu kreieren oder Daten zu präsentieren und identifizieren. Die Sprache verwendet dabei Tags, um Elemente in einem Dokument zu definieren beziehungsweise zu beschreiben. Die Datei wird bei der Implementierung dabei in zwei verschiedenen Teilen aufgeteilt. Der Header beeinhaltet die Metadaten, den Titel und weiteres. Im Body werden alle Elemente beschrieben, welche sichtbar sein sollen.<sup>13</sup>

#### **Unterschiede: Auszeichnungssprachen - Programmiersprachen**

Eine Programmiersprache wird grundsätzlich durch einen Compiler oder Interpreter ausgeführt. Auszeichnungssprachen werden vom Browser interpretiert. Bei der Verwendung gibt es einige Unterschiede. Auszeichnungssprachen werden verwendet, um Daten zu präsentieren beziehungsweise darzustellen. Programmiersprachen hingegen geben Befehle an den Computer. Der Computer führt anhand dieser Befehle und Anweisungen eigenständig Prozesse durch. Beispiele für Programmiersprachen sind Python, C, C++, Java und weitere. Beispiele für Auszeichnungssprachen sind HTML, XML und weitere.<sup>13</sup>

---

<sup>13</sup>Quelle: Vgl. - 17.01.2021, <https://pediaa.com/what-is-the-difference-between-markup-language-and-programming-language/>

---

### 3.2.8 HTML, CSS und JS

#### HTML

HTML ist eine Markup-Language - also eine Auszeichnungssprache. HTML steht für "Hyper Text Markup Language". Hyper Text bedeutet soviel wie "Text in einem Text". Ein Text, in dem ein Link enthalten ist, ist also Hypertext. Immer wenn auf einen Link geklickt wird, welcher zu einer Webseite führt, wird gleichzeitig auch auf Hypertext geklickt. Markup Language wurde unter 3.2.7 behandelt. Mit HTML können grundlegende statische Webseiten erstellt werden. Ein HTML Dokument welches korrekt geschrieben wurde läuft problemlos auf jedem Browser, und das Betriebssystemunabhängig.<sup>14</sup>

#### CSS

CSS steht für "Cascade Styling Sheets". Es ist eine Stylesheet-Sprache und Sie wird dafür verwendet um das Aussehen und die Formatierung eines Dokumentes zu beschreiben, welches in einer Markup Language (zum Beispiel in HTML) geschrieben wurde. CSS kann dabei das Layout mehrerer Webseiten gleichzeitig strukturieren und manipulieren. Es wird zusammen mit HTML und JavaScript in den meisten Websites verwendet, um Benutzeroberflächen für Webanwendungen und Benutzeroberflächen für mobile Anwendungen zu erstellen.<sup>15</sup>

#### JS

JavaScript ist eine Programmiersprache für das Web. Es kann sowohl HTML als auch CSS aktualisieren und ändern. JavaScript kann Daten berechnen, manipulieren und validieren. Mittlerweile hat sich JavaScript zusammen mit CSS und HTML als Standard-Technologie durchgesetzt und etabliert. Wenn eine statische Webseite dynamisiert werden soll, verwendet man dafür JavaScript.<sup>16</sup>

---

<sup>14</sup>Quelle: Vgl. - 17.01.2021, <https://www.javatpoint.com/what-is-html>

<sup>15</sup>Quelle: Vgl. - 17.01.2021, <https://www.javatpoint.com/what-is-css>

<sup>16</sup>Quelle: Vgl. - 17.01.2021, [https://www.w3schools.com/whatis/whatis\\_js.asp](https://www.w3schools.com/whatis/whatis_js.asp)

---

### 3.2.9 Datenbanken

Eine Datenbank ist eine Sammlung von strukturierten Informationen oder Daten, die in einem Computersystem gespeichert werden. Sie wird normalerweise von einem Datenbankmanagementsystem (DBMS) gesteuert. Gemeinsam werden die Daten und das DBMS, zusammen mit den Anwendungen, die mit ihnen verbunden sind, als Datenbanksystem bezeichnet, oft auch nur als Datenbank abgekürzt. Die Umsetzung einer Datenbank erfolgt, für gewöhnlich, in Form einer Tabellensammlung. Jede Tabelle hat dabei Zeilen und Spalten. Somit wird der Datenfluss effizient und unkompliziert durchgeführt. Daten einer Datenbank, beziehungsweise eines DMBS, können somit ziemlich einfach verwaltet, geändert, aktualisiert, kontrolliert und organisiert werden. Ein Großteil der Datenbanken wird dabei mit einer "structured query language" (SQL) umgesetzt. Diese wird für das Schreiben und Abfragen von Daten verwendet. Es gibt verschiedene Typen von Datenbanken wie, relationale Datenbanken, objektorientierte Datenbanken und weitere.<sup>17</sup>

### 3.2.10 SQLite

SQLite ist eine C-Sprachbibliothek die ein relationales Datenbankmanagementsystem mitliefert. Dabei wird eine kleine, schnelle, in sich geschlossene, hochzuverlässige und voll funktionsfähige SQL-Datenbank-Engine implementiert. SQLite ist die meistgenutzte Datenbank-Engine der Welt. Die Datenbank läuft dabei unabhängig und serverlos. SQLite braucht dabei keinen Serverprozess, sie ist direkt in der Applikation, welche auf die Datenbank zugreift, integriert (im Gegenteil zu MySQL oder PostgreSQL). Das SQLite-Dateiformat ist stabil, plattformübergreifend und abwärtskompatibel. Die Tabellen, welche in SQLite erstellt werden können sind dynamisch. Es ist also jeden Datentyp in jedes Feld der Datenbank problemlos lagern. Der SQLite-Quellcode ist in der Public-Domain und kann von jedem für jeden Zweck frei verwendet werden.<sup>18</sup>

---

<sup>17</sup>Quelle: Vgl. - 17.01.2021, <https://www.oracle.com/database/what-is-database/>

<sup>18</sup>Quelle: Vgl. - 17.01.2021, <https://www.sqlite.org/index.html>

---

## 3.3 Grundlagen: Netzwerk

### 3.3.1 7-Schichten-OSI-Modell

OSI steht für “Open System Interconnection“ und es ist ein Referenzmodell. Es beschreibt den Informationsfluss von einer Softwareanwendung eines Computers, über ein Transportmedium, zu der Softwareanwendung eines anderen Computers. Dabei wird das OSI-Modell in sieben verschiedene Schichten unterteilt. Jede Schicht hat eine maßgeschneiderte Funktion im Gesamtsystem. Alle sieben Schichten ergeben das “Big Picture“.

#### Schicht 1 - Bitübertragungsschicht (Physical Layer)

Die unterste Schicht des OSI-Modells befasst sich mit der elektrischen und physischen Darstellung des Systems. Dies können Netzwerkkabel, Radiofrequenz (Bei 802.11 WLAN) oder auch andere Systemeigenschaften sein. Ein Beispiel für ein Layer 1 Problem wäre ein Lan-Kabel, welches nicht korrekt verbunden ist.<sup>19</sup> & <sup>20</sup>

#### Schicht 2 - Sicherungsschicht (Data Link Layer)

Die Sicherungsschicht ist grundsätzlich in zwei Unterschichten aufgebaut. Die erste Schicht ist Media Access Control (MAC) Layer. Hier wird Flusskontrolle und Multiplexing für Geräteübertragungen durchgeführt. Die meisten Switches arbeiten auf Layer 2, diese sind jedoch nicht für virtuelle Local Area Networks zu verwenden. Dafür gibt es aber Layer 3 Switches.<sup>19</sup> & <sup>20</sup>

#### Schicht 3 - Netzwerkschicht (Network Layer)

Auf der Netzwerkschicht finden so gut wie alle Routerfunktionalitäten und Routerprozesse statt. Frames werden hier aufgefangen und je nach der, in Frames stehenden Adresse, weitervermittelt. Die Ziele werden durch IP-Adressen definiert und wiedergefunden. Router spielen bei der Weiterleitung von Datenpaketen die ausschlaggebende Rolle.<sup>19</sup> & <sup>20</sup>

#### Schicht 4 - Transportschicht (Transport Layer)

Diese Schicht koordiniert den Datentransfer zwischen Systemen und Hosts. Das Transfer Control Protocol (TCP) baut auf Internet Protocol (IP) auf und findet daher auf dieser Schicht den vollen Nutzen.<sup>19</sup> & <sup>20</sup>

---

<sup>19</sup> Quelle: Vgl. - 17.01.2021, <https://bit.ly/3w1OpEI> (Link mit bitly.com gekürzt)

<sup>20</sup> Quelle: Vgl. - 17.01.2021, <https://www.forcepoint.com/de/cyber-edu/osi-model>

---

## **Schicht 5 - Sitzungsschicht (Session Layer)**

Wenn zwei Computer miteinander kommunizieren wollen, muss eine Session erstellt werden. Die Sitzungsschicht erstellt, verwaltet und steuert diese Session beziehungsweise Sitzung. So mit gehört zu dieser Schicht das Setup also das Erstellen neuer Sitzungen. Außerdem gehören auch die Koordination und die Termination zu Schicht 5. Die Koordination beschreibt beispielsweise die Systemwartezzeit für gewisse Antworten. Die Termination beendet die Verbindung nach Ende der Session.<sup>19</sup> & <sup>20</sup>

## **Schicht 6 - Darstellungsschicht (Presentation Layer)**

Die sechste Schicht stellt grob gesagt einen Dolmetscher zwischen einer Anwendung und dem Netzwerk. Auf dieser Schicht werden Daten formatiert und dann an die Anwendungsschicht weitergegeben. Datenverschlüsselung und Entschlüsselung findet auch auf dieser Schicht statt. Die Darstellungsschicht wird oftmals auch Syntaxschicht genannt.<sup>19</sup> & <sup>20</sup>

## **Schicht 7 - Anwendungsschicht (Application Layer)**

Die höchste hierarchisch höchste Schicht ist die Anwendungsschicht. Diese Schicht befindet, ist für Endnutzer am greifbarsten. Die siebte Schicht stellt Netzwerkdienste zur Verfügung, welche eine Kommunikation zwischen zwei Endnutzern ermöglichen. Beispiele für Schicht-sieben Applikationen sind Web-Browser oder Programme wie Skype. Die Endnutzer interagieren so mit direkt mit den auf der Anwendungsschicht basierenden Programmen.<sup>19</sup> & <sup>20</sup>

---

### **3.3.2 IPv4-Adressen**

IPv4 Adressen sind 32-Bit lange Zahlenkombinationen, welche in zwei Segmente aufgeteilt sind. Geräte in einem Netzwerk werden zur Identifizierung und zum Abbild einer Netzwerk-hierarchie mit IP-Adressen versehen. Der erste Teil einer IP-Adresse ist der Network Präfix, der zweite Teil die Host-Nummer. Jedes am Internet beteiligte Gerät hat eine individuelle IP-Adresse, anhand welcher das Gerät auffindbar ist. Die IP-Adresse spielt beim Fernzugriff eine ausschlaggebende Rolle.<sup>21</sup>

### **3.3.3 Portweiterleitung**

Portweiterleitung ist der Prozess, bei dem Datenverkehr auf eine IP- beziehungsweise Port-Kombination zugeschickt, dort abgefangen und anschließend an einen anderen Port umgeleitet wird. Portweiterleitung wird auch für das HSS verwendet, um den Fernzugriff zu ermöglichen (Implementierung des Fernzugsgriffs - siehe 5.4). Dabei verweist der Router auf die IP-Adresse des Raspberry Pi. Auf dem Raspberry Pi läuft dabei der Webserver und somit kann global auf diesen zugegriffen werden.<sup>22</sup>

### **3.3.4 SSL-Zertifikate**

Webzertifikate werden für gewöhnlich dazu verwendet um als Server, dem Client eine verschlüsselte Verbindung zu garantieren (Erstellen und Implementieren des Web-Zertifikates - siehe 5.3.2.6). Wenn der Client eine Anfrage erstellt, sollte demnach der Webserver mit einem SSL-Zertifikat antworten. Der Client kann dabei anschließend nach einer Überprüfung, das Zertifikat als vertrauenswürdig symbolisieren. Nach der Verifizierung wird vom Client selbst ein Encryption-Key erstellt, durch welchen die darauffolgende Kommunikation mit dem Server gesichert ablaufen kann.<sup>23</sup>

---

<sup>21</sup>Quelle: Vgl. - 19.01.2021, <https://juni.pr/3flSRIu> (Link mit bitly.com gekürzt)

<sup>22</sup>Quelle: Vgl. - 19.01.2021, <https://de.wizcase.com/blog/portweiterleitung-was-ist-es/>

<sup>23</sup>Quelle: Vgl. - 19.01.2021, <https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https>

---

## 3.4 Grundlagen: 3D-Druck

Die folgenden Informationen wurden aus dem 3D-Druck Laborvorwissen der 5. Klasse (Herr Prof. Hackl Wolfgang), der offiziellen AutoDesk-Webseite, von ProtoTec und von StrataSys entnommen.

### 3.4.1 Definition des 3D-Drucks

„3D-Druck, auch als additive Fertigung bezeichnet, ist eine Kombination von Prozessen zur Fertigung von Objekten durch Hinzufügen von Material in Schichten, die den aufeinanderfolgenden Querschnitten eines 3D-Modells entsprechen. Kunststoffe und Metalllegierungen werden am häufigsten für den 3D-Druck verwendet. Es ist aber praktisch jedes Material geeignet – von Beton bis zu lebendem Gewebe.“<sup>24</sup>

### 3.4.2 Materialkunde

Für verschiedene Anwendungsbereiche müssen auch verschiedene Materialien, Oberflächen oder Farben benutzt werden.

#### 3.4.2.1 Materialien

- **ASA - Acrylnitril-Styrol-Acrylat**<sup>25</sup>
  - Einsatzgebiet: für Teile im Außenbereich
  - Eigenschaften: UV- und Witterungsbeständig
- **ABS - Acrylnitril-Butadien-Styrol**<sup>25</sup>
  - Einsatzgebiet: für technische Teile
  - Eigenschaften: Schlagfest, hohe Festigkeit
- **PET/PETG - Polyethylenterephthalat**<sup>25</sup>
  - Einsatzgebiet: für transparente Modelle, Konsumgüter
  - Eigenschaften: Transluzent, Lebensmitteleignung

---

<sup>24</sup>Quelle: Vgl - 22.03.2021, <https://www.autodesk.de/solutions/3d-printing>

<sup>25</sup>Quelle: Vgl - 22.03.2021, <https://www.prototec.de/3d-druck-materialien>

- 
- **PC (Polycarbonate)**<sup>25</sup>
    - Einsatzgebiet: für transparente und steife Modelle
    - Eigenschaften: Transluzent, hohe Steifigkeit
  - **PU - Polyurethan**<sup>25</sup>
    - Einsatzgebiet: Elastische Bauteile
    - Eigenschaften: gummiartig
  - **PEEK/ULTEM1010**<sup>25</sup>
    - Einsatzgebiet: Teile für spezielle Anwendungen
    - Eigenschaften: Hochtemperatur/chemische Beständigkeit
  - **PLA - Polylactide**<sup>25</sup>
    - Einsatzgebiet: einfache Modelle
    - Eigenschaften: Günstig, einfach zu drucken

### 3.4.2.2 Verwendung im Projekt

Für dieses Projekt wurde ABS, genauer gesagt ABS M30, verwendet, da es mehrheitlich im technischen Segment verwendet wird und es sehr stoßfest und sehr schlagfest agiert. Die gute thermische Beständigkeit (-20 Grad Celsius bis 110 Grad Celsius) ist ein Vorteil für dieses Projekt, da ein Gehäuse gebaut wird und die innere Wärmentwicklung eine Rolle spielt. Die Anwendung dieses Materials sind technische Teile wie zum Beispiel Apparatebau, Armaturenbau, Anlagenbau oder Maschinenbau.<sup>26</sup>

„ABS-M30 ist 25 bis 70 Prozent stärker als Standard-ABS und eignet sich daher ideal für die Konzeptstudien, das funktionale Prototyping, Fertigungswerzeuge und Endanwendungen. Mit deutlich stärkerer Schichthaftung als ABS und höherer Zug-, Schlag- und Biegefestigkeit sind ABS-M30-Teile stärker, glatter und weisen bessere Detailmerkmale auf.“<sup>27</sup>

---

<sup>26</sup>Quelle: Vgl. - 22.03.2021, <https://www.prototec.de/wp-content/uploads/2021/01/PROTOTEC-3D-Druck-Datenblatt-ABS-FDM.pdf>

<sup>27</sup>Quelle: 22.03.2021, <https://www.stratasys.com/de/materials/search/abs-m30>

---

## 4 WAHL DER ARBEITSMEDIEN

### 4.1 Hardwareauswahl

#### 4.1.1 Raspberry 3b Plus

Für dieses Projekt wurde, der schon von der HTL Anichstraße für den Unterricht benutzte, Raspberry 3b Plus verwendet. Da die GPIO Pins und andere Ports einfach benutzbar sind und das Pi Interface (nachdem das Betriebssystem erfolgreich installiert wurde) sehr einfach zu verstehen ist, fiel die Entscheidung schnell auf das Raspberry Pi 3 und nicht auf einen Microcontroller wie zum Beispiel den Arduino. Die für dieses Projekt benötigten Slots beziehungsweise Ports sind unten aufgelistet. (Haftungsausschluss: Eine Benutzung dieses Systems wird nur mit der Vergewisserung der Benutzung aller wichtigen (3) Ports und Slots garantiert. Es darf keine Schnittstelle ausgelassen werden.)

- **Local Area Network Port**

Dieser Port wird für die Kommunikation mit dem Laptop verwendet. Der Laptop dient zur Steuerung des Raspberry Pi Systems (via VNC Connection). Sie dient auch als Netzwerkbrücke. Die LAN Verbindung muss daher immer garantiert werden. Eine Wireless Implementierung des Projektes wurde nicht berücksichtigt.

- **Micro USB Port**

Für die Spannungsversorgung muss der Raspberry Pi 3b Plus mit einem Micro USB Kabel mit dem Laptop verbunden werden. Der Laptop stellt eine USB 2.0 5V Spannungsversorgung bereit. Dies ist genau die benötigte Spannung, die der Raspberry Pi zum Funktionieren braucht.

- **Micro SD Karten-Slot**

Die 16 GB große SD Karte wird mit dem Betriebssystem überschrieben. Es wird das offizielle Image Flash Programm "Raspberry Pi OS Imager" benutzt. Dies hat sich in der Vergangenheit als sehr zuverlässig und vertrauenswürdig bewiesen. Neben dem Betriebssystem werden alle benötigten Programme auch auf der SD Karte gespeichert.

## 4.1.2 Zusätzliche Komponenten

#### 4.1.2.1 Kamera

Die abgebildete Abbildung zeigt die eingebaute Komponente. Es muss beachtet werden, dass hier die Pi-Revision 1.3 benutzt wurde. Sie bietet eine farbintensive Bilderfassung und flüssige Datenübertragungen. (Haftungsausschluss: Dieses Projekt wurde nicht mit der anderen Cam-Revision (Rev 2.1) getestet.)<sup>28</sup>

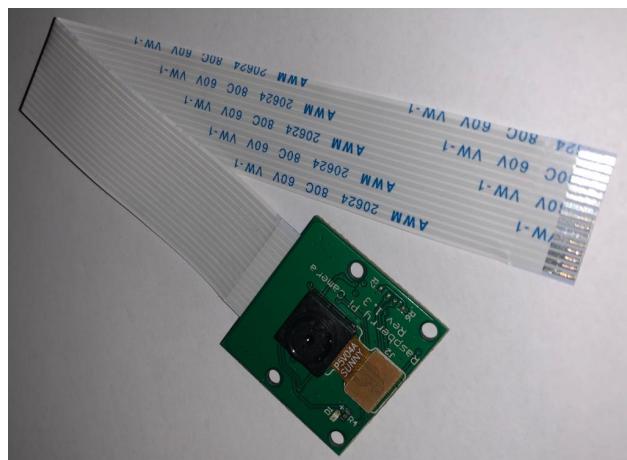


Abbildung 9: Pi Cam Rev 1.3

Für die Live-Videoüberwachung im Webserver wird die standardmäßig bereitgestellte Raspberry Pi Kamera benutzt. Die Dimensionen der Kamera sind 25mm x 20mm x 9mm. Der Sensor selbst hat eine Bildauflösung von 5 Megapixel mit einem Fixfokusobjektiv. Das Fixfokusobjektiv wird bei einfachen, lichtschwachen Kameras beziehungsweise Linsen benutzt, um auf eine Entfernungseinstellung zu verzichten. Der von dieser Kamera erfasste Eventbereich beträgt ungefähr 250 Centimeter. Diese Pi Cam wird auch gewählt, da der Einkaufsspreis gegenüber anderen Kameras sehr angemessen ist. Das erfasste Bild wird am Webserver dargestellt und kann LIVE (teilweise Verzögerungen von maximal 750ms erkennbar) betrachtet werden. Es werden durchschnittlich 30 Bilder pro Sekunde übertragen und die maximale Auflösung liegt bei 1080p (= Full-HD).<sup>28</sup>

<sup>28</sup>Quelle: Vgl - 28.02.2021, <https://docs.rs-online.com/2888/0900766b8127db0a.pdf>

---

#### 4.1.2.2 Motion-Sensor

Als Bewegungserkennung beziehungsweise Motion Detection wird der HC-SR501 Sensor benutzt. Es ist ein (Passiver) Pyroelektrischer Infrarot Sensor, auch genannt PPIR oder PIR. Dieses Bauteil ist deshalb als passiv anzusehen, weil der Bewegungsmelder selbst keine Daten oder Signale aussendet. Es dient lediglich zum Empfang der Daten innerhalb des Evenhorizonts - dieser liegt bei genau 110 Grad. Es besitzt 3 Pin (Datenausgang + Spannungsversorgung).<sup>29</sup>

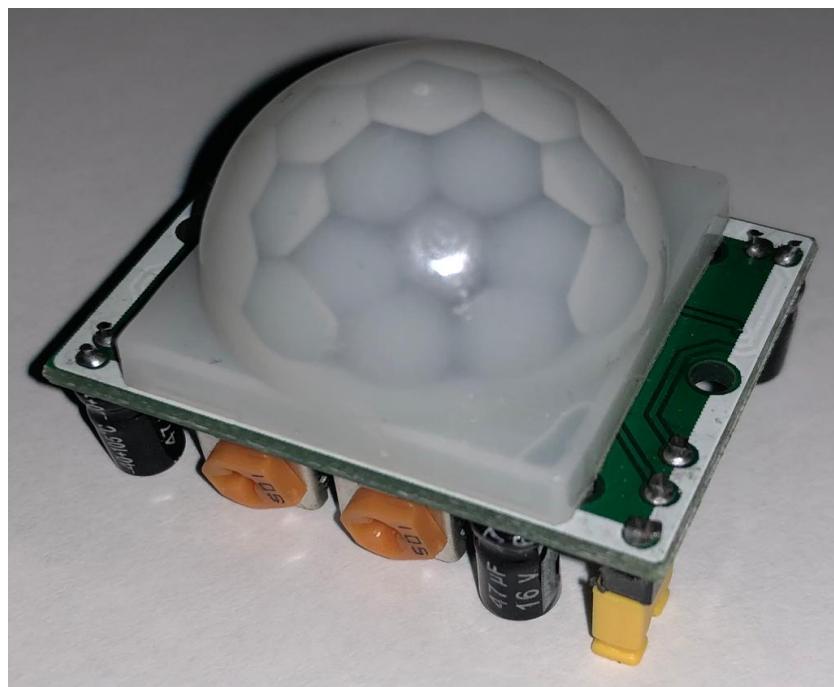


Abbildung 10: HS-SR501 Bewegungsmelder

---

<sup>29</sup>Quelle: Vgl - 28.02.2021, <https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html>

---

#### 4.1.2.3 Temperatur- und Luftfeuchtigkeitssensor

Als Temperatur- und Luftfeuchtigkeitssensor wird der bekannte DHT11 Sensor (Abbildung 6) benutzt. Dieser Sensor ist mit Abstand der günstigste unter Temperatur- und Luftfeuchtigkeitssensoren. Dies war auch einer der Gründe für die Auswahl dieser Komponente. Der DHT11 Sensor besitzt vier Pins (Datenausgang + Spannungsversorgung + Not Connected) und die Platine besitzt 3 Pins (Datenausgang + Spannungsversorgung). (Haftungsausschluss: Das System wird nur für die Verwendung für Temperaturen über 0 Grad Celsius bis 50 Grad Celsius empfohlen, da Messungen unter 0 Grad Celsius oder über 50 Grad Celsius mit dem DHT11 Sensor technisch nicht möglich sind)<sup>30</sup>

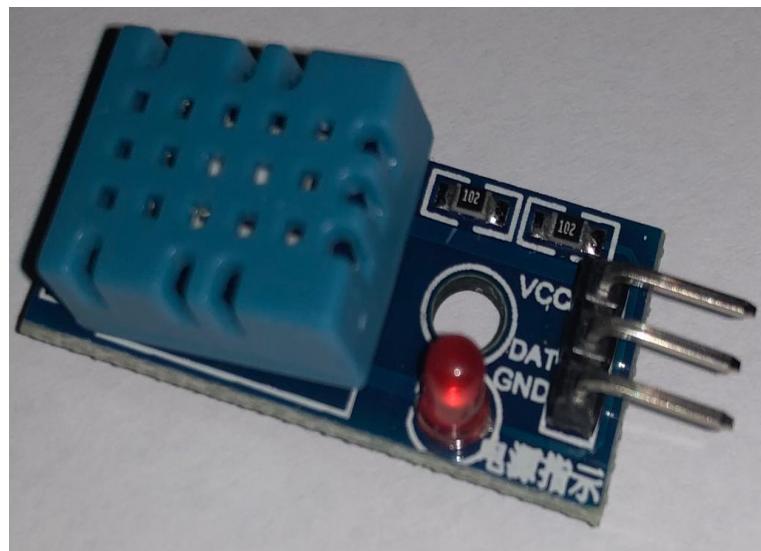


Abbildung 11: DHT 11 Sensor mit Outputplatine

---

<sup>30</sup>Quelle: Vgl. - 28.02.2021, <https://www.alldatasheet.com/datasheet-pdf/pdf/1132088/ETC2/DHT11.html>

---

## 4.2 Softwareauswahl

### 4.2.1 Programmiersprachen und Frameworks

#### 4.2.1.1 Python

Python wird in diesem Projekt als Hauptprogrammiersprache verwendet. Das Hauptprogramm, sowie einzelne Unterprogramme, zur Ansteuerung diverser Komponenten, werden in Python 3 Programmiert. Python 3 wurde gewählt, weil damit ziemlich unkompliziert IoT-Projekte konzipiert werden können. Außerdem wird Raspbian (das Betriebssystem, welches auf dem Raspberry Pi 3B Plus läuft) mit Python, der offiziellen Programmiersprache, ausgeliefert. In diesem Fall war der Raspberry Pi schon mit Python 3 ausgestattet.

#### 4.2.1.2 Flask

Nach der Recherche, ob für den Webserver Flask oder Django als Web Framework verwendet werden soll, fiel die Auswahl auf Flask. Der Grund dafür war unter anderem die Unkompliziertheit von Flask. In Flask kann zum Beispiel eine Hello-World Seite mit geringerem Aufwand aufgebaut werden.

Weitere Vorteile von Flask (welche in diesem Projekt ausschlaggebend waren) sind folgende.<sup>31</sup>

- „Leichtere Handhabung für einfache Fälle“<sup>31</sup>
- „Die Codebasis ist relativ klein (im Vergleich zu Django)“<sup>31</sup>
- „Routing URL ist einfach integrierbar“<sup>31</sup>
- „Datenbankintegration ist unkompliziert“<sup>31</sup>

#### 4.2.1.3 SQLite

Zur Datenbankauswahl standen zwei nutzbare Optionen: SQLite und MySQL. MySQL ist sehr bekannt, doch zu pompös und aufwendig (außerdem gehört es zu Oracle) für den Einsatz auf einfachen Raspberry-basierten Projekten. SQLite ist wahrscheinlich die am besten geeignete Wahl, denn es ist serverlos, leichtgewichtig, Open-Source und unterstützt den meisten SQL-Code.

---

<sup>31</sup>Quelle: 28.02.2021, <https://www.guru99.com/flask-vs-django.html>

---

#### **4.2.1.4 HTML, CSS und JS**

Hypertext Markup Language (**HTML**), Cascading Style Sheets (**CSS**) und Java Script (**JS**) werden als Basis für den Webserver verwendet.

### **4.2.2 Weitere Software, Plugins, Frameworks und Libraries**

#### **4.2.2.1 Adafruit DHT Sensor Library - *Python Library***

Diese Library von Adafruit wird verwendet um den DHT11 Sensor zu konfigurieren.

#### **4.2.2.2 Matplotlib - *Python Library***

Die Matplotlib Library wird verwendet um die, in der Datenbank abgespeicherten Werte, grafisch abzubilden.

#### **4.2.2.3 jQuery - *JavaScript Library***

Aus der jQuery-Library werden Plugins für Styling-Zwecke verwendet.

#### **4.2.2.4 Raphaël - *JavaScript Library***

Wird zusätzlich für JustGage in das Programm implementiert.

#### **4.2.2.5 JustGage - *JavaScript Plugin***

Das JavaScript Plugin, JustGage, wird für die Veranschaulichung, der aktuell gemessenen Temperatur- und Luftfeuchtigkeitswerte, verwendet. Es basiert auf der Raphaël-Library für Vektorzeichnungen

#### **4.2.2.6 Bootstrap - *CSS Framework***

Das Bootstrap-Framework wird für Styling-Zwecke verwendet. Dazu werden die CDN-Links für Bootstrap-Stylesheets und jQuery-Plugins in beide Dateien aufgenommen.

#### **4.2.2.7 OpenCV - *freie Library***

Diese freie Library wird zur flüssigen Videoübertragung der Pi Cam Rev 1.3 genutzt.

---

## 4.3 Basis

### 4.3.1 Erklärung

Auf GitHub wurde ein ähnliches Projekt gesichtet. Das GitHub-Projekt stellt eine perfekte Basis für die Implementierung des HSS dar.<sup>32</sup>

Das Problem beim Weiterverwenden fremder Codes ist die Lizenz. In diesem Fall ist zum Code noch eine MIT-license hinterlegt, die wie folgt aussieht:

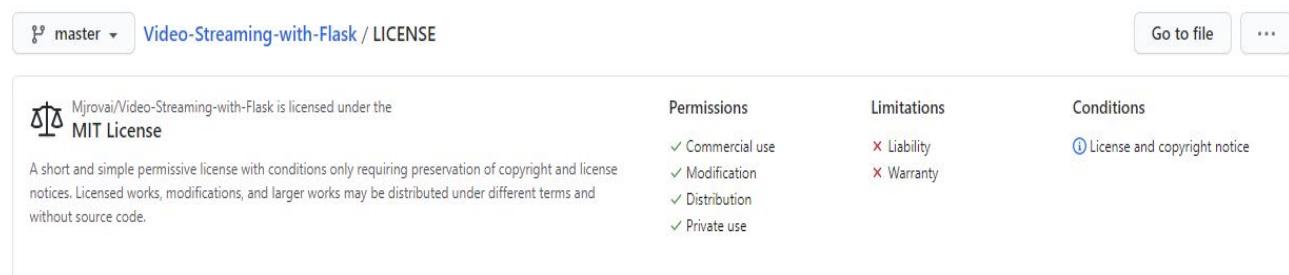


Abbildung 12: Basis: Lizenz

Wie in der Abbildung zu erkennen ist, erlaubt die Lizenz folgende Verwendungszwecke:

- Kommerzielle Benutzung
- Änderung
- Verteilung beziehungsweise Aufteilung
- privater Gebrauch

Um sicherzustellen, dass zusätzlich keine Urheberrechte verletzt werden, wurde ein Mailkontakt mit dem Autor hergestellt. Nach einer kurzen Weile bestätigte der Autor mit einer Antwortmail, dass dieses Projekt weiterverwendet werden darf. Somit ist dieses GitHub-Projekt frei zu gebrauchen und zum öffentlichen Nachbau geeignet. Das GitHub-Projekt wird als Basis für die softwaretechnische Implementierung des Home-Security-Systems verwendet. Die genauen beziehungsweise aufgelisteten Verwendungszwecke können unter 4.3.2 eingesehen werden.

---

<sup>32</sup>Quelle: 22.03.2021, <https://github.com/Mjrovai/Video-Streaming-with-Flask>

---

#### 4.3.2 Verwendungszwecke und Unterschiede

Die folgende Tabelle beschreibt die Verwendungszwecke der GitHub Basis und die Unterschiede bei der Implementierung des HSS.

Projektbereich	GitHub-Basis	HSS (Home Security System)
Temp.- u. Luftfeuchtigkeitssensor	DHT 22	DHT 11
Kamera Programmimplementierung Dafür verwendete Software	PI NoIR Night Camera Eigenes Programm Pi-camera-module	PI Cam Rev 1.3 im Hauptprogramm eingebunden OpenCV Library zur Optimierung
Bewegungssensor Programmimplementierung Zusätzlich	kein Bewegungssensor keine Implementierung	PIR Motion Sensor implementiert Eigenes Programm Beifügen der Ausgabe im Kamerabild
Datenbank für DHT Sensor	Implementiert in SQLite	Entnommen aus Basis
Hauptprogramm	Implementiert in Python	Entnommen aus Basis und erweitert. - Redesign der Matplotlib-Diagramme - Korrektur des Kamerafehlers (siehe 5.2.2.5) - Optimierung des Kamerastreams (OpenCV) - Hinzufügen des Motion-Sensors-Outputs - Einbau des Fernzugriffs
Fernzugriff	kein Fernzugriff	Mittels Port-Forwarding umgesetzt. Erstellung eines Webzertifikats mit cert.pem und key.pem als Zertifikatsdateien.
index.html	Implementiert	Entnommen aus Basis und erweitert. - Redesign der kompletten Front-Page - Implementierung der Buttons mit Bootstrap
kamera.html	camera.html implementiert	Entnommen aus Basis und erweitert. - Implementierung mit Bootstrap - Redisgn der Page
style.css	Implementiert	Eigenständige Implementierung

Tabelle 2: Basis: Verwendungszwecke und Unterschiede

---

## 5 PRAKТИСЧЕ УМСЕТЗУНГ

### 5.1 Vorbereitung des Raspberry Pi 3B Plus

In diesem Kapitel wird die Erstaufsetzung beziehungsweise das Flashen der SD-Karte mit dem Betriebssystem vorgeführt. Dieser Vorgang wird in der Regel nicht mehr als 30 Minuten beanspruchen. Für die Installation ist eine SD-Karte, eine aktive Internetverbindung und natürlich der Raspberry Pi von nötig. Zum Raspberry Pi System kann eine VNC-Verbindung hergestellt werden - bei dieser Installation wurde jedoch auf eine HDMI-Verbindung mit einem Bildschirm gesetzt.

#### 5.1.1 Flashen der SD-Karte mit Raspberry Pi OS

#### Schritt 1: OS Imager Downloaden und Installieren



Abbildung 13: imageresetup.exe ist gestartet

Nachdem der OS Imager von der offiziellen Raspberry Pi Seite '<https://www.raspberrypi.org/>' gedownloadet wurde, wird die imageresetup.exe Datei gestartet. Nachdem man auf 'Install' klickt, wird der Raspberry Pi Imager installiert.

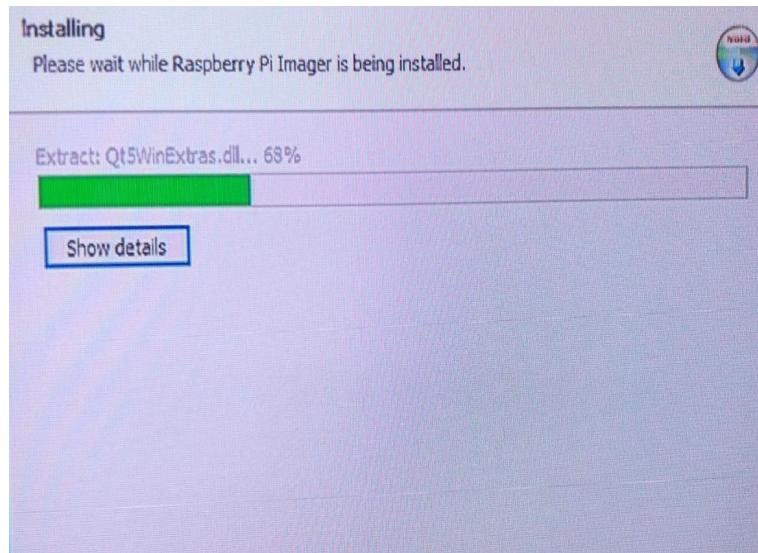


Abbildung 14: Raspberry OS Imager wird installiert

Dieser Vorgang benötigt in der Regel nicht mehr als einer Minute.

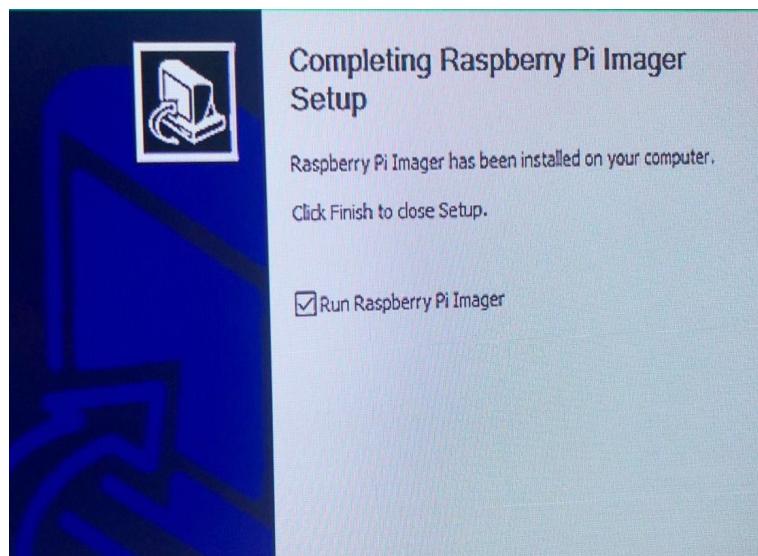


Abbildung 15: Raspberry OS Imager ist installiert

Nachdem die Installation beendet wurde, kann man das Setup-Programm (imagersetup.exe) schließen.

---

## Schritt 2: OS auf SD-Karte Flashen



Abbildung 16: OS Imager Dashboard

Der OS Imager ist in drei Bereiche unterteilt:

- Auswahl des Betriebssystems
- Auswahl der SD-Karte
- Flash-Befehl

---

Als Erstes wird das Betriebssystem ausgewählt:

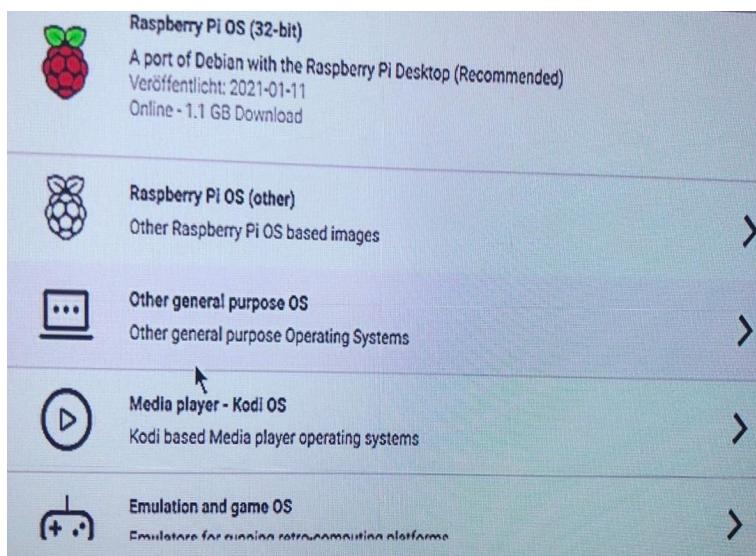


Abbildung 17: OS Imager Dashboard: OS Auswahl

Dieses Projekt ist für die Raspberry Pi OS Version 2021-01-11 konzipiert. Mit einem Klick auf das Betriebssystem ist diese ausgewählt.

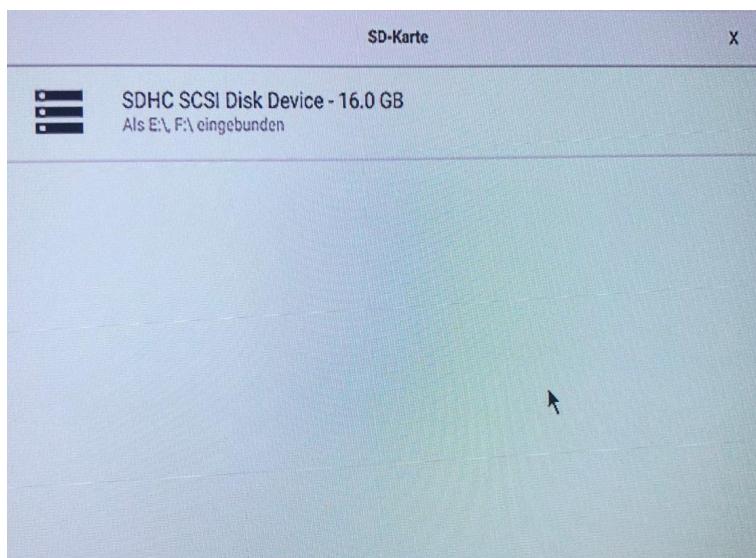


Abbildung 18: OS Imager Dashboard: SD-Karten Auswahl

Hier wählt man die, zu flashende, SD-Karte. Sie sollte groß genug sein, dass die Programme, die neben des Betriebssystems auch auf der SD-Karte gespeichert werden, genug Speicherplatz zur Verfügung haben.

---

Sobald alles, OS und SD-Karte, ausgewählt wurde, klickt man auf 'Schreiben' und die SD-Karte wird mit dem ausgewähltem Betriebssystem überschrieben beziehungsweise geflashed.



Abbildung 19: OS Imager Dashboard: Flashen beendet

Nachdem Flashen entfernt man die SD-Karte, fügt diese in den Raspberry Pi SD-Karten Slot ein und bootet den Raspberry Pi mit der neu-aufbereiteten SD-Karte.



Abbildung 20: Raspberry Pi: Erfolgreiche Installation des Betriebssystems

Nach der Betriebssystem-Installation, sieht man eine Willkommensnachricht.

---

## Schritt 3: Setup des Raspberry Pi



Abbildung 21: Raspberry Pi: Betriebssystem-Setup gestartet

Ab hier werden personalisierte Einstellungen durchgeführt

Personalisiertes Setup beinhaltet:

- Einrichtung eines Passworts
- Wahl des Landes und Sprache
- Einrichtung einer WLAN-Verbindung

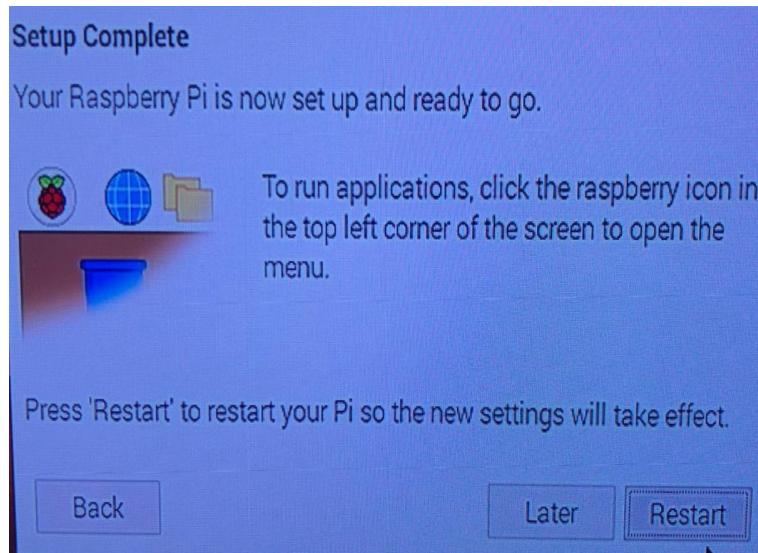


Abbildung 22: Raspberry Pi: Betriebssystem-Setup fertig

Nachdem Aufsetzen des Raspberry Pi kann man das System noch neu starten, um den Bootvorgang zu testen.

### 5.1.2 Erstellung des Programmverzeichnisses

Auf dem Raspberry Pi Desktop wird ein Ordner namens "Diplomarbeit" erstellt. Dort werden alle für dieses Projekt umgesetzten Programme gespeichert.

---

## 5.2 Software- und Hardwaretechnische Umsetzung der Bauteile

### 5.2.1 PIR Motion Sensor

Das Programm wird in den Hauptordner gespeichert (siehe 5.1.2.).

#### 5.2.1.1 Grundlegende Funktionsbeschreibung

Der PIR Motion Sensor soll menschliche Bewegungen beziehungsweise Lichtveränderungen im Sensorbereich erkennen, um diese anschließend auf der Live-Videoübertragung des Webserver anzulegen. Dabei wird der Bewegungssensor in das Gehäuse des HSS eingebaut und mit dem Raspberry Pi verbunden.

#### 5.2.1.2 Hardwaretechnische Umsetzung

Für den Bewegungssensor werden drei Pins des Raspberry Pi benötigt. Zur Versorgung wird der Pin 4 (+5V) verwendet. Für die Datenübertragung wird der Pin 16 (GPIO 23) verwendet. Als GND wird Pin 6 (GND) verwendet.

#### 5.2.1.3 Überblick aller Programmfunctionen

Zur Übersichtlichkeit und Verständlichkeit werden alle Funktionen des Programms tabellarisch dargestellt. Dabei stellt die linke Spalte, die Initialisierung der Funktionen dar und die rechte Spalte beschreibt die jeweiligen Funktionen.

Initialisierung der Funktion	Funktionsbeschreibung
def bewegungScannen():	Tastet bei Ausführung Pin 16 ab und gibt, je nach Resultat, entsprechende Strings zurück.

Tabelle 3: PIR Motion Sensor: Programmfunction

---

#### 5.2.1.4 Softwaretechnische Implementierung

##### [ imports und "Vorkonfiguration"]:

Bevor Funktionen implementiert werden, müssen Libraries eingebunden werden. Außerdem muss der Pin 16 (GPIO 23) konfiguriert werden. All dies wird in Python implementiert.

---

```
1 import RPi.GPIO as GPIO
2 import time
3
4 #Es wird der Raspberry Pi Pin 16 (GPIO 23) konfiguriert
5 GPIO.setwarnings(False)
6 GPIO.setmode(GPIO.BOARD)
7 GPIO.setup(16, GPIO.IN)
```

---

Listing 1: PIR Sensor: imports und Vorkonfiguration

##### [ def bewegungScannen() ]:

Die Funktion ist in Python implementiert. Die Funktion liest den Pin 16 (GPIO 23), an welchem der Datenpin des PIR Motion Sensor angeschlossen ist, ein. Anschließend gibt die Funktion, je nach Ergebnis (1 oder 0), "Bewegung" oder "Keine Bewegung" als Strings zurück.

Die Implementierung sieht wie folgt aus:

---

```
1 def bewegungScannen():
2     #wenn Bewegung stattfindet gibt dies Funktion "Bewegung" zurück,
3     #ansonsten gibt diese Funktion "Keine Bewegung" zurück.
4     if ((GPIO.input(16)==1):
5         return "Bewegung"
6     else:
7         return "Keine Bewegung"
```

---

Listing 2: PIR Sensor: bewegungScannen()

---

## 5.2.2 Kamera

### 5.2.2.1 Grundlegende Funktionsbeschreibung

Die Kamera wird zur Liveübertragung des Webservers genutzt. Dabei wird diese in das Gehäuse des HSS eingebaut und mit dem Raspberry PI verbunden.

### 5.2.2.2 Hardwaretechnische Umsetzung

Die Kamera wird über einem Flachband CSI Kabel (15 Bit) mit dem CSI camera connector des Raspberry PI verbunden.

### 5.2.2.3 Installation der Kamera

Bevor die Kamera verwendet werden kann, muss diese in den Raspberry PI Einstellungen aktiviert werden. Dafür navigiert man zu Preferences -> Raspberry Pi Configuration (ersichtlich in der folgenden Abbildung)



Abbildung 23: PI Cam Installation: Navigierung zum Konfigurationsmenü

---

Anschließend kann die Kamera im Submenü “Interfaces“, unter “Camera“, aktiviert werden. (ersichtlich in der folgenden Abbildung)

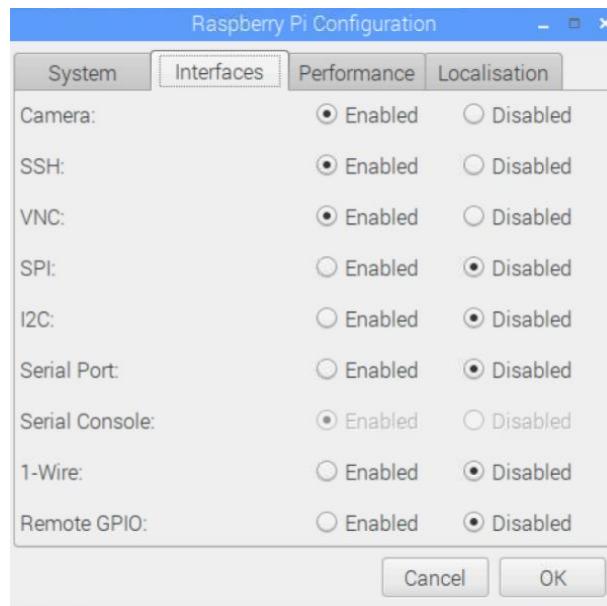


Abbildung 24: PI Cam Installation: Aktivierung der Kamera

#### 5.2.2.4 Installation der OpenCV Library

Die OpenCV Library wird zur Liveübertragung über die PI Cam verwendet. Die Installation der Library wird im Raspberry Pi Terminal ausgeführt und erfolgt dabei über die Raspbian Repositories:

Als Erstes wird geupdatet:

---

```
$ sudo apt update
```

---

Anschließend kann OpenCV über folgenden Befehl installiert werden:

---

```
$ sudo apt install python3-opencv
```

---

Nun kann OpenCV in Python-Programmen verwendet werden.

---

#### 5.2.2.5 Das Kameraproblem und die Lösung:

Die Kamera wird im Hauptprogramm (siehe 5.3.2.5 / kameraGenerator()) implementiert. Das Problem, welches sich im Laufe des Projekts ereignet hat, betrifft den Einbau der Kamera in das Gehäuse. Die Bohrungen für die Kamera wurden 180-Grad verdreht umgesetzt. Daraus resultierend entstand ein neues Problem. Die Textausgabe des Bewegungssensors wurde zwar korrekt angezeigt, jedoch war die Videoübertragung dabei um 180-Grad, genau wie die Bohrungen, verdreht. Dieses Problem wurde softwaretechnisch gelöst, indem der Bildwinkel, welcher von der Kamera aufgenommen wird, zuerst mit folgender OpenCV Funktion korrigiert wurde.

---

```
#Aufnahme des Frames
ret, frame = kameraAufnahme.read()

#Rotierung des Frames um das Kameraproblem zu beheben beziehungsweise zu umgehen.
frameMirror = cv2.rotate(frame, cv2.ROTATE_180)
```

---

Anschließend kann der Text des Bewegungssensors mit folgender Funktion in das Bild korrekt implementiert werden.

---

```
#Einfügen des Motion-Sensor Textes in das Frame
frameMirror = cv2.putText(frameMirror, bewegungScannen(),(50,50),cv2
    .FONT_HERSHEY_SIMPLEX, 1,(0,0,0), 1, cv2.LINE_AA, False)
```

---

---

### 5.2.3 Datenbank

#### 5.2.3.1 Grundlegende Funktionsbeschreibung

Der DHT11 Sensor liest Messwerte aus, welche anschließend vom Programm in die SQLite Datenbank gespeichert werden. Das Hauptprogramm greift außerdem auf die Datenbank zurück um, mittels der Messwerte auf der Webseite, das Monitoring zu erledigen.

#### 5.2.3.2 Installation von SQLite

Die Datenbank basiert auf SQLite 3. SQLite 3 kann im Raspberry Pi Terminal installiert werden. Dafür wird folgender Befehl ausgeführt:

---

```
$ sudo apt-get install sqlite3
```

---

#### 5.2.3.3 Softwaretechnische Ausarbeitung der Datenbank

Im Raspberry Pi Terminal kann die Datenbank umgesetzt werden. Alternativ kann dafür auch Python verwendet werden. Die Datenbank wird in den Ordner /Diplomarbeit erstellt, welcher auf dem Desktop liegt (siehe 5.1.2). Dafür navigiert man mittels folgendem Befehl zum Ordner:

---

```
$ cd Desktop/Diplomarbeit
```

---

Anschließend kann mittels folgendem Befehl die Datenbank erstellt werden:

---

```
$ sqlite3 Datenbank.db
```

---

Nun befindet man sich in der SQLite Shell. Hier wird die zu nutzende Tabelle erstellt. Die Tabelle umfasst drei Spalten. Die erste Spalte namens "zeit" verwendet den Datentyp DATETIME. Die zweite Spalte namens "temperatur" verwendet den Datentyp NUMERIC. Die dritte Spalte namens "luftfeuchtigkeit" verwendet den Datentyp NUMERIC.

---

```
> BEGIN;  
> CREATE TABLE Daten_Tabelle (zeit DATETIME, temperatur NUMERIC, luftfeuchtigkeit NUMERIC);  
> COMMIT;
```

---

---

So wird die Tabelle getestet:

```
sqlite> INSERT INTO Daten_Tabelle VALUES(datetime('now'), 27.5, 31);
sqlite> INSERT INTO Daten_Tabelle VALUES(datetime('now'), 26.5, 30);
sqlite> INSERT INTO Daten_Tabelle VALUES(datetime('now'), 22.1, 21);
sqlite> SELECT * FROM Daten_Tabelle;
2021-03-04 18:59:23|27.5|31
2021-03-04 18:59:32|26.5|30
2021-03-04 18:59:42|22.1|21
sqlite> █
```

Abbildung 25: SQLite3 Datenbank: Test der Tabelle in der Datenbank

#### **“INSERT INTO ... VALUES“**

Hier werden vom Benutzer willkürliche Werte in die dazugehörigen Spalten eingefügt.

#### **“SELECT \* FROM ...“**

Ausgabe jeglicher Daten in der Tabelle “Daten\_Tabelle“.

In diesem Fall wurden die Daten richtig eingefügt und ausgegeben.

⇒ Die Datenbank funktioniert.

---

## 5.2.4 DHT 11

### 5.2.4.1 Grundlegende Funktionsbeschreibung

Der DHT 11 Sensor misst die Umgebungstemperatur (Ausgabe in Grad Celsius) und die Luftfeuchtigkeit (Ausgabe in Prozent). Das Programm speichert die Werte in die Tabelle "Daten\_Tabelle" der Datenbank.

### 5.2.4.2 Hardwaretechnische Umsetzung

Für den DHT 11 Sensor werden drei Pins des Raspberry Pi benötigt. Zur Versorgung wird der Pin 1 (+3.3V) verwendet. Für die Datenübertragung wird der Pin 36 (GPIO 16) verwendet. Als Ground wird Pin 34 (GND) verwendet. Als Letztes wird ein Vorwiderstand (am besten 5 bis 10k Ohm) zwischen Pin 1 und Pin 36 geschalten, welcher den DHT 11 schützt. Was ohne diesen Widerstand passiert, kann man in 6.1 sehen.

### 5.2.4.3 Installation der ADAFRUIT Library

Auf GitHub ist eine zum DHT Sensor dazugehörige Library veröffentlicht, die in diesem Projekt Anwendung findet. Der GitHub Ordner wird gedownloadet und in den zuvor erstellten Ordner (siehe 5.1.2) abgelegt, wo sich die bisher erstellten Programme und die Datenbank befinden.<sup>33</sup>

Als Nächstes kann die Library installiert werden. Dazu navigiert man im Raspberry Pi Terminal zum Ordner, welcher von GitHub heruntergeladen wurde und führt anschließend die Datei "setup.py" aus.

---

```
$ cd Desktop/Diplomarbeit/Adafruit_Python_DHT-master/
$ sudo python3 setup.py install
```

---

Nun kann der Ordner "Adafruit\_Python\_DHT-master" aus dem "Diplomarbeit" Ordner entfernt werden. Die Library ist jetzt installiert und mittels Python verwendbar.

---

<sup>33</sup>Quelle: 10.03.2021, [https://github.com/adafruit/Adafruit\\_Python\\_DHT](https://github.com/adafruit/Adafruit_Python_DHT)

---

#### 5.2.4.4 DHT 11 Test

Zu Testzwecken, ob zum Beispiel der Sensor richtig funktioniert, wird ein kleines Testprogramm in Python geschrieben. Das Programm importiert die Adafruit Library und initialisiert die GPIO Pins des Raspberry Pi. Anschließend misst der DHT Sensor die Temperatur und Luftfeuchtigkeit. Als Letztes gibt der Sensor diese Werte am Terminal beziehungsweise in der Shell aus.

---

```
1 import Adafruit_DHT
2 luftfeuchtigkeit, temperatur = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 16)
3 print('Temperatur={0:0.1f}*C Luftfeuchtigkeit={1:0.1f}%''
4     .format(temperatur, luftfeuchtigkeit))
```

---

Listing 3: DHT 11: Testprogramm

Die Ausgabe des Programmes sieht wie folgt aus:



```
Shell
>>>
>>> %Run DHT_Sensor_Test.py
Temperatur=26.0*C Luftfeuchtigkeit=33.0%
```

Abbildung 26: DHT 11 Sensor: Testprogramm Ausgabe

Wie in der oben dargestellten Abbildung gesehen werden kann, funktioniert der DHT 11 Sensor beziehungsweise auch die Adafruit Library.

---

#### 5.2.4.5 Überblick aller Programmfunctionen

Zur Übersichtlichkeit und Verständlichkeit werden alle Funktionen des Programms tabellarisch dargestellt. Dabei stellt die linke Spalte die Initialisierung der Funktionen dar und die rechte Spalte beschreibt die jeweiligen Funktionen.

Initialisierung der Funktion	Funktionsbeschreibung
def datenSpeichern (temperatur, luftfeuchtigkeit):	Speichert die, als Parameter gegebenen Werte, in die Datenbank ab.
def datenAnzeigen():	Nimmt die, in der Datenbank gespeicherten Werte und gibt diese aus.
def datenAuslesen():	Initialisiert den GPIO-Pin des Raspberry Pi, liest Daten aus und nutzt weitere Funktionen.
def main():	Nutzt Funktionen in einer gewissen Zeitperiode.

Tabelle 4: DHT 11: Programmfunctionen

#### 5.2.4.6 Softwaretechnische Ausarbeitung aller Programmfunctionen

[ imports ]:

Das Programm wird in den Ordner Diplomarbeit gespeichert (siehe 5.1.2). Bevor Funktionen implementiert werden, müssen Libraries eingebunden werden. (SQLite, Adafruit und time)

---

```
1 import time
2 import sqlite3
3 import Adafruit_DHT
```

---

Listing 4: DHT 11: imports

---

### [ def datenSpeichern (temperatur, luftfeuchtigkeit) ]:

Die Funktion ist in Python implementiert. Sie hat zwei Parameter. Diese stehen für die gemessene Temperatur und Luftfeuchtigkeit des DHT 11 Sensors.

---

```
1 def datenSpeichern (temperatur, luftfeuchtigkeit):
2
3     #Es wird eine Verbindung zur SQLite Datenbank hergestellt.
4     conn=sqlite3.connect('Datenbank.db')
5     curs=conn.cursor()
6
7     #Anschließend werden, die über Parameter übergebenen Daten, mittels
8     #INSERT INTO Befehl in die SQLite Datenbank eingespeichert.
9     curs.execute("INSERT INTO Daten_Tabelle values(datetime('now','localtime'),
10     (?,?,?,?,?), (temperatur, luftfeuchtigkeit))
11
12     #Die Transaktion wird nun beendet.
13     conn.commit()
14     conn.close()
```

---

Listing 5: DHT 11: datenSpeichern(temperatur, luftfeuchtigkeit)

### [ def datenAnzeigen() ]:

Die Funktion ist in Python implementiert. Sie hat keine Parameter. Die Funktion ist durch Kommentare beschrieben und sieht wie folgt aus:

---

```
1 def datenAnzeigen():
2     #Es wird eine Verbindung zur SQLite Datenbank hergestellt.
3     conn = sqlite3.connect('Datenbank.db')
4     curs = conn.cursor()
5
6     #Anschließend werden die Werte mittels SELECT Befehl aus der Tabelle geholt
7     #und dann mittels print Befehl ausgegeben.
8     for row in curs.execute("SELECT * FROM Daten_Tabelle ORDER BY zeit DESC LIMIT 1"):
9         zeit = str(row[0])
10        temperatur = row[1]
11        luftfeuchtigkeit = row[2]
12        print(zeit,temperatur,luftfeuchtigkeit)
```

---

Listing 6: DHT 11: datenAnzeigen()

---

### [ def datenAuslesen() ]:

Die Funktion ist in Python implementiert. Sie hat keine Parameter. Die Funktion ist durch Kommentare beschrieben und sieht implementiert wie folgt aus:

---

```
1 def datenAuslesen():
2     #Parameter 1 der folgenden Funktion, "Adafruit_DHT.DHT11" ruft den DHT11 Sensor auf
3     #Parameter 2 der folgenden Funktion, "16" steht für den GPIO 16 beziehungsweise
4     #Pin 36 des Raspberry Pi
5     luftfeuchtigkeit, temperatur = Adafruit_DHT.read_retry(Adafruit_DHT.DHT11, 16)
6
7     #wenn Werte ausgelesen werden konnten,
8     #wird datenSpeichern() und datenAnzeigen() aufgerufen.
9     if luftfeuchtigkeit is not None and temperatur is not None:
10         luftfeuchtigkeit = round(luftfeuchtigkeit)
11         temperatur = round(temperatur, 1)
12         datenSpeichern(temperatur, luftfeuchtigkeit)
13         datenAnzeigen()
```

---

Listing 7: DHT 11: datenAuslesen()

### [ def main() ]:

Die Funktion ist in Python implementiert. Sie hat keine Parameter, da dies die main-function ist. Die Funktion führt den kompletten Prozess des DHT 11 aus.

---

```
1 #Bei Ausführung der Main-Funktion wird in Dauerschleife
2 #datenAuslesen() aufgerufen. Dabei wird eine Zeitverzögerung von 60 sekunden
3 #eingebaut. Diese ist dazu da, um die gewünschte Messfrequenz von einer Messung
4 #pro Minute zu liefern.
5 def main():
6     while True:
7         datenAuslesen()
8         time.sleep(60)
```

---

Listing 8: DHT 11: main()

---

## 5.3 Hauptprogramm

### 5.3.1 Aufsetzen der Ordnerstruktur

Zu Beginn der praktischen Umsetzung wurde auf dem Raspberry Pi Desktop ein Ordner namens Diplomarbeit erstellt. (siehe 5.1.2)

Bisher wurden dort folgende Dateien eingebaut:

- **Bewegungs\_Sensor.py** (Das Programm für den PIR Motion Sensor)
- **DHT\_Sensor.py** (Das Programm für den DHT 11 Sensor)
- **Datenbank.db** (Die Datenbank für das Programm)

#### Der nächste Schritt:

Als nächstes müssen für das Hauptprogramm vier weitere Dateien erstellt werden.

- **main.py** (Die Datei sollte sich im Hauptordner “Diplomarbeit” befinden. )
- **style.css** (Für diese Datei wird ein neuer Ordner im “Diplomarbeit-Ordner“ namens “static“ erstellt. )
- **index.html** (Für diese Datei wird ein neuer Ordner im “Diplomarbeit-Ordner“ namens “templates“ erstellt. )
- **kamera.html** (Auch diese Datei sollte im Ordner “templates“ gespeichert werden. )

Die Dateistruktur sieht demnach wie folgt aus:

```
Desktop
  > Diplomarbeit
    > Bewegungs_Sensor.py
    > DHT_Sensor.py
    > main.py
    > templates
      > index.html
      > kamera.html
    > static
      > style.css
```

Abbildung 27: Hauptprogramm: Ordnerstruktur

---

### 5.3.2 Backend

#### 5.3.2.1 Grundlegende Funktionsbeschreibung

Das Hauptprogramm vereint das Frontend mit dem Backend. Außerdem wird hier die Kamera implementiert und das Programm befindet sich im Ordner “Diplomarbeit“. (siehe 5.1.2 oder 5.3.1)

#### 5.3.2.2 Überblick aller Programmfunctionen

Zur Übersichtlichkeit und Verständlichkeit werden alle Funktionen des Programms tabellarisch dargestellt. Dabei stellt die linke Spalte die Initialisierung der Funktionen dar und die rechte Spalte beschreibt die jeweiligen Funktionen.

Initialisierung der Funktion	Funktionsbeschreibung
def auslesenLetzterDaten():	Liest die letzten Daten der Datenbank aus
def pruefeGrenzen(temperaturen, luftfeuchtigkeiten):	Prüft, ob sich die gemessenen Werte, in einem, vorher definiertem Bereich, befinden.
def auslesenHistorischerDaten (anzahlAbtastungen):	Liest die letzten X Daten aus der Datenbank aus. X ist hierbei der Funktionsparameter.
def anzahlReihen():	Gibt die Anzahl der Reihen in der Datenbank zurück.
def abtastungsPeriode():	Errechnet die Abtastungsperiode des DHT 11 Sensors und gibt diese zurück.
def index():	Stellt die Front-Page des Webservers dar.
def my_form_post():	Liefert Seitenaktualisierung, sobald historische Werte gefordert werden.
def tempearturdiagramm():	Nutzt historische Werte, um ein Liniendiagramm, in Form einer PNG zu erstellen und zurückzugeben.
def luftfeuchtigkeitsdiagramm():	Nutzt historische Werte, um ein Liniendiagramm, in Form einer PNG zu erstellen und zurückzugeben.
def kameraseiteUpdate():	Aktualisiert die Kameraseite
def kameraGenerator():	Die Hauptfunktion der Kamera
def liveuebertragung():	Gibt die Funktion “kameraGenerator“ zurück.

Tabelle 5: Hauptprogramm: Programmfunctionen

---

### 5.3.2.3 Installation von Matplotlib

Matplotlib wird, durch Befehle im Raspberry Pi Terminal, von GitHub heruntergeladen. Das Klonen von GitHub geschieht dabei mittels folgendem Befehl:

---

```
$ git clone https://github.com/matplotlib/matplotlib
```

---

Anschließend wird zum heruntergeladenen Ordner beziehungsweise Verzeichnis navigiert.

---

```
$ cd matplotlib
```

---

Nun wird das setup.py aufgebaut und anschließend installiert.

---

```
$ python3 setup.py build  
$ sudo python3 setup.py install
```

---

### 5.3.2.4 Installation von Flask

Als Nächstes wird Flask, das Web-Framework, welches für dieses Projekt benutzt wurde, installiert. Dafür wird vorerst folgender Befehl ausgeführt.

---

```
$ sudo apt-get install python3-flask
```

---

---

### 5.3.2.5 Softwaretechnische Implementierung

Das Hauptprogramm wird in Python implementiert. Dabei wird Flask als Hauptframework verwendet.

#### [ imports ]:

Bevor Funktionen implementiert werden, müssen Libraries eingebunden werden. Außerdem müssen die bereits erstellten Programme eingebunden werden.

```
1 #Bewegungs_Sensor.py ist das Programm, welches für den PIR Motion Sensor geschrieben wurde.  
2 #daraus wird die Funktion bewegungScannen() importiert.  
3 from Bewegungs_Sensor import bewegungScannen  
4  
5 #DHT_Sensor ist das Programm welches für den DHT 11 Sensor geschrieben wurde.  
6 #daraus wird die Funktion main() importiert  
7 from DHT_Sensor import main  
8  
9 #Die restlichen Libraries und Frameworks werden hier importiert.  
10 import threading  
11 import cv2  
12 from datetime import datetime  
13 from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas  
14 from matplotlib.figure import Figure  
15 import io  
16 from flask import Flask, render_template, send_file, make_response, request, Response  
17 import sqlite3
```

---

Listing 9: Hauptprogramm: imports

---

## [ Initialisierungen ]:

Bevor Funktionen implementiert werden, müssen außerdem folgende Elemente initialisiert werden.

---

```
1 #Initialisierung der Flask App
2 app = Flask(__name__)
3
4 #Initialisierung der SQLite Datenbank
5 conn=sqlite3.connect('~/Datenbank.db')
6 curs=conn.cursor()
```

---

Listing 10: Hauptprogramm: Initialisierungen

## [ auslesenLetzterDaten() ]:

Die Funktion wird wie folgt implementiert.

---

```
1 def auslesenLetzterDaten():
2     #Auslesen des letzten Eintrages der Datenbank, für alle drei Spalten,
3     #durch den SQLite Query. Der LIMIT Befehl limitiert das Auslesen auf einen Eintrag,
4     #somit wird nicht die komplette Tabelle ausgelesen, sondern nur die letzte Reihe.
5     for spalte in curs.execute("SELECT * FROM Daten_Tabelle ORDER BY zeit DESC LIMIT 1"):
6         zeit = str(spalte[0])
7         temperatur = spalte[1]
8         luftfeuchtigkeit = spalte[2]
9
10    #Anschließendes Zurückgeben der Daten.
11    return zeit, temperatur, luftfeuchtigkeit
```

---

Listing 11: Hauptprogramm: auslesenLetzterDaten()

---

### [ pruefeGrenzen(temperaturen, luftfeuchtigkeiten) ]:

Die Funktion verwendet die gemessene Temperatur und Luftfeuchtigkeit als Parameter. Es wird geprüft, ob die Werte sich im gewünschten Bereich befinden. Die Funktion wird wie folgt implementiert.

```
1 def pruefeGrenzen(temperaturen, luftfeuchtigkeiten):
2     #Es wird die Länge der Liste "temperaturen" geholt.
3     laenge = len(temperaturen)
4
5     #Anschließend wird jeder Temperatur- und Luftfeuchtigkeitswert auf
6     #die gewünschten Bereiche geprüft.
7     #Für Temperatur: 0°C bis 50°C, da dies der Bereich ist, in dem der
8     #DHT 11 Sensor noch einigermaßen plausible Werte messen kann.
9     #Für Luftfeuchtigkeit werden alle Werte zwischen 0% und 100%
10    #als plausibel angenommen.
11
12    #Sollten die Werte sich im ungewünschten Bereich befinden, so wird
13    #dies als Messfehler interpretiert und der Messwert wird auf den, in der
14    #Datenbank vorherigen Messwert gesetzt.
15
16    #Dies dient zur Garantierung, dass auch bei Messfehlern ein sauberes
17    #grafisches Monitoring durchgeführt werden kann.
18    for i in range(0, laenge-1):
19        if (temperaturen[i] < 0 or temperaturen[i] >50):
20            temperaturen[i] = temperaturen[i-1]
21        if (luftfeuchtigkeiten[i] < 0 or luftfeuchtigkeiten[i] >100):
22            luftfeuchtigkeiten[i] = luftfeuchtigkeiten[i-1]
23
24    #Zuletzt erfolgt noch eine Rückgabe der Werte.
25    return temperaturen, luftfeuchtigkeiten
```

---

Listing 12: Hauptprogramm: pruefeGrenzen(temperaturen, luftfeuchtigkeiten)

---

## [ auslesenHistorischerDaten(anzahlAbtastungen) ]:

Die Funktion wird wie folgt implementiert.

```
1 def auslesenHistorischerDaten(anzahlAbtastungen):
2     #Es werden historische Daten ausgelesen.
3     #"+str(anzahlAbtastungen))" bestimmt dabei wieviele Reihen (zeitlich
4     #absteigend) insgesamt ausgelesen werden.
5     curs.execute("SELECT * FROM Daten_Tabelle ORDER BY zeit DESC LIMIT "
6                 +str(anzahlAbtastungen))
7
8     #alle ausgelesenen Daten werden auf die Variable daten zwischengespeichert.
9     daten = curs.fetchall()
10
11    #Initialisierung von Listen für die jeweiligen Spalten der Datentabelle.
12    zeiten = []
13    temperaturen = []
14    luftfeuchtigkeiten = []
15
16    #Befüllen der Listen mit ausgelesenen Daten.
17    for spalte in reversed(daten):
18        zeiten.append(spalte[0])
19        temperaturen.append(spalte[1])
20        luftfeuchtigkeiten.append(spalte[2])
21
22    #Prüfung, ob sich die Werte im gewünschten Bereich befinden.
23    temperaturen, luftfeuchtigkeiten = pruefeGrenzen(temperaturen, luftfeuchtigkeiten)
24
25    #Rückgabe der Werte.
26    return zeiten, temperaturen, luftfeuchtigkeiten
```

---

Listing 13: Hauptprogramm: auslesenHistorischerDaten(anzahlAbtastungen)

---

### [ **anzahlReihen()** ]:

Die Funktion wird wie folgt implementiert.

---

```
1 def anzahlReihen():
2     #Für die Temperaturspalte wird nachgezählt, wieviele Einträge vorhanden sind.
3     for spalte in curs.execute("select COUNT(temperatur) from Daten_Tabelle"):
4         #Dies wird in die Variable reihenAnzahl eingeschrieben.
5         reihenAnzahl=spalte[0]
6
7     #Anschließend wird die Variable zurückgegeben.
8     return reihenAnzahl
```

---

Listing 14: Hauptprogramm: anzahlReihen()

### [ **abtastungsPeriode()** ]:

Die Funktion errechnet den Zeitabstand zwischen zwei Einträgen in die Datenbank. Damit kann geprüft werden, ob die gewünschte Abtastungsperiode des DHT 11 Sensors eingehalten wird.

---

```
1 def abtastungsPeriode():
2     #Mit der, in Listing 13, erklärten Funktion werden historische Daten ausgelesen.
3     zeiten, temperaturen, luftfeuchtigkeiten = auslesenHistorischerDaten (2)
4
5     #Anschließend wird die Variable "zeiteinheit" initialisiert.
6     zeiteinheit = '%Y-%m-%d %H:%M:%S'
7
8     #Periode 0 und Periode 1 werden definiert und anschließend voneinander subtrahiert.
9     periode0 = datetime.strptime(zeiten[0], zeiteinheit)
10    periode1 = datetime.strptime(zeiten[1], zeiteinheit)
11    abtastung = periode1-periode0
12
13    #Die Abtastungsperiode wird auf Sekunden heruntergerechnet und zurückgegeben.
14    abtastung = int(round(abtastung.total_seconds()/60))
15    return (abtastung)
```

---

Listing 15: Hauptprogramm: abtastungsPeriode()

---

## [ Globalisierung bedeutender Variablen ]:

Um Variablen funktionsübergreifend verwenden zu können, werden diese globalisiert.

```
1 #Es wird geprüft, ob bereits 60 oder mehr Einträge in der Datenbank vorhanden sind.  
2 #Falls dies zutrifft, wird diese Variable auf 60 gesetzt.  
3 global anzahlAbtastungen  
4 anzahlAbtastungen = anzahlReihen()  
5 if (anzahlAbtastungen >= 60):  
6     anzahlAbtastungen = 60  
7  
8 #Die Abtastungsperiode wird, mithilfe der in Listing 15  
9 #implementierten Funktion, initialisiert.  
10 global abtastungsPerioden  
11 abtastungsPerioden = abtastungsPeriode()  
12  
13 #Dies ist die Variable, welche bestimmt, wieviele historische Werte ausgelesen  
14 #und anschließend grafisch dargestellt werden.  
15 global anzahlGeforderterWerte  
16 anzahlGeforderterWerte = 15
```

---

Listing 16: Hauptprogramm: Globalisierung bedeutender Variablen

---

## [ index() ]:

Die Funktion wird wie folgt implementiert.

```
1 #Erstellen des Programmthreads
2 programmThread = threading.Thread(target = main)
3
4 #Flask Initialisierung der Front-Page.
5 @app.route("/")
6 def index():
7     #Auslesen der letzten gemessenen Daten mittels auslesenLetzterDaten(),
8     #siehe Listing 11.
9     zeit, temperatur, luftfeuchtigkeit = auslesenLetzterDaten()
10
11    #Erstellen einer Variable "datenVorlage" zur Datenausgabe.
12    #datenVorlage wird dabei mit den vorher geholten Werten befüllt.
13    datenVorlage = {
14        'zeit' : zeit,
15        'temperatur' : temperatur,
16        'luftfeuchtigkeit' : luftfeuchtigkeit,
17        'abtastung' : abtastungsPerioden,
18        'anzahlGeforderterWerte' : anzahlGeforderterWerte
19    }
20
21    #Abfrage, ob Programmthread schon läuft. Falls nicht, wird dieser gestartet.
22    if not programmThread.is_alive():
23        programmThread.start()
24
25    #Rendern der HTML-Page und Übergabe von "datenVorlage" an die HTML-Page.
26    return render_template('index.html', **datenVorlage)
```

---

Listing 17: Hauptprogramm: index()

---

## [ my\_form\_post() ]:

Die Funktion wird wie folgt implementiert.

```
1 #Diese Funktion verwendet die HTTP-POST Methode. Die Methode dient dazu,
2 #Daten an den Server zu senden beziehungsweise die Webseite zu aktualisieren.
3 @app.route('/', methods=['POST'])
4 def my_form_post():
5     #Initialisierung der bereits globalisierten verfügbaren Variablen
6     global anzahlAbtastungen
7     global abtastungsPerioden
8     global anzahlGeforderterWerte
9
10    #Auslesen der, vom User geforderten, Anzahl an darzustellender historischen Werte.
11    anzahlGeforderterWerte = int (request.form['anzahlGeforderterWerte'])
12
13    #Wenn die Abtastungsperiode des DHT Sensors 2 Minuten ist und der User
14    #als "anzahlGeforderterWerte" 1 eingibt entsteht ein Fehler. Dieser wird hier
15    #beobauen beziehungsweise wird es hiermit verhindert.
16    if (anzahlGeforderterWerte < abtastungsPerioden):
17        anzahlGeforderterWerte = abtastungsPerioden + 1
18
19    #Errechnung der Anzahl der Abtastungen auf Basis der geforderten Werte
20    #und der realen abtastungsPeriode des DHT 11 Sensors.
21    anzahlAbtastungen = anzahlGeforderterWerte//abtastungsPerioden
22    numMaxPerioden = anzahlReihen()
23
24    #Schutz vor dem Fall, dass der User mehr Werte darstellen will, als in der
25    #Datenbank verfügbar.
26    if (anzahlAbtastungen > numMaxPerioden):
27        anzahlAbtastungen = (numMaxPerioden-1)
28
29    #Auslesen letzter Daten
30    zeit, temperatur, luftfeuchtigkeit = auslesenLetzterDaten()
31
32    #Erstellen einer Datenvorlage für das Frontend
33    datenVorlage = {
34        'zeit' : zeit,
35        'temperatur' : temperatur,
36        'luftfeuchtigkeit' : luftfeuchtigkeit,
37        'abtastung' : abtastungsPerioden,
38        'anzahlGeforderterWerte' : anzahlGeforderterWerte
39    }
40
41    #Rendern der HTML-Page und Übergabe von "datenVorlage" an die HTML-Page.
42    return render_template('index.html', **datenVorlage)
```

---

Listing 18: Hauptprogramm: my\_form\_post()

---

## [ temperaturdiagramm() ]:

Die Funktion wird wie folgt implementiert.

```
1 #Diese Funktion erstellt eine PNG-Datei beziehungsweise ein Abbild eines, in Matplotlib
2 #erstellten, Temperaturdiagrammes.
3 @app.route('/plot/temperaturdiagramm')
4 def temperaturdiagramm():
5     #Auslesen und Initialisieren historischer Daten aus der Datenbank.
6     zeiten, temperaturen, luftfeuchtigkeiten = auslesenHistorischerDaten(anzahlAbtastungen)
7
8     #Konfiguration des zu erstellenden Diagrammes.
9     ys = temperaturen
10    fig = Figure(facecolor="#181818")
11    axis = fig.add_subplot(1, 1, 1, axisbg="#181818")
12    axis.spines['top'].set_color('white')
13    axis.spines['bottom'].set_color('white')
14    axis.spines['left'].set_color('white')
15    axis.spines['right'].set_color('white')
16    axis.set_title("Temperatur [°C]", color = 'white')
17    axis.set_xlabel("Zeit", color = 'white')
18    axis.set_ylabel("Temperatur", color = 'white')
19    axis.tick_params(axis='x', colors='white')
20    axis.tick_params(axis='y', colors='white')
21    axis.grid(True)
22
23    #Beifügen der Zeitachse (x-Achse) zum Diagramm.
24    zeit = []
25    for l in zeiten:
26        zeit.append(".".join(l.split(" ")[1].split(":")[:slice(2)]))
27    xs = zeit
28
29    #Plotten beziehungsweise Erstellen des Diagrammes.
30    axis.plot(xs, ys, linewidth = 5)
31    canvas = FigureCanvas(fig)
32    output = io.BytesIO()
33    canvas.print_png(output)
34
35    #Rückgabe des Diagrammes an den Webserver
36    ausgabe = make_response(output.getvalue())
37    ausgabe.mimetype = 'image/png'
38    return ausgabe
```

---

Listing 19: Hauptprogramm: temperaturdiagramm()

---

## [ luftfeuchtigkeitsdiagramm() ]:

Die Funktion wird wie folgt implementiert. Der Code ähnelt stark Listing 19, da beide Funktionen eigentlich das selbe Problem (nur ein wenig abgeändert) lösen müssen.

---

```
1 #Diese Funktion erstellt eine PNG-Datei beziehungsweise ein Abbild eines in Matplotlib
2 #erstellten Luftfeuchtigkeitsdiagrammes.
3 @app.route('/plot/luftfeuchtigkeitsdiagramm')
4 def luftfeuchtigkeitsdiagramm():
5     #Auslesen und Initialisieren historischer Daten aus der Datenbank.
6     zeiten, temperaturen, luftfeuchtigkeiten = auslesenHistorischerDaten(anzahlAbtastungen)
7
8     #Konfiguration des zu erstellenden Diagrammes.
9     ys = luftfeuchtigkeiten
10    fig = Figure(facecolor="#181818")
11    axis = fig.add_subplot(1, 1, 1, axisbg="#181818")
12    axis.spines['top'].set_color('white')
13    axis.spines['bottom'].set_color('white')
14    axis.spines['left'].set_color('white')
15    axis.spines['right'].set_color('white')
16    axis.set_title("Luftfeuchtigkeit [%]", color = 'white')
17    axis.set_xlabel("Zeit", color = 'white')
18    axis.set_ylabel("Luftfeuchtigkeit", color = 'white')
19    axis.tick_params(axis='x', colors='white')
20    axis.tick_params(axis='y', colors='white')
21    axis.grid(True)
22
23    #Beifügen der Zeitachse (x-Achse) zum Diagramm.
24    zeit = []
25    for l in zeiten:
26        zeit.append(".".join(l.split(" ")[1].split(":")[:slice(2)]))
27    xs = zeit
28
29    #Plotten beziehungsweise Erstellen des Diagrammes.
30    axis.plot(xs, ys, linewidth = 5)
31    canvas = FigureCanvas(fig)
32    output = io.BytesIO()
33    canvas.print_png(output)
34
35    #Rückgabe des Diagrammes an den Webserver
36    ausgabe = make_response(output.getvalue())
37    ausgabe.mimetype = 'image/png'
38    return ausgabe
```

---

Listing 20: Hauptprogramm: luftfeuchtigkeitsdiagramm()

---

## [ kameraseiteUpdate() ]:

```
1 #Diese Funktion updated die Kamera-Seite.  
2 @app.route('/kamera')  
3 def kameraseiteUpdate():  
4     return render_template('kamera.html')
```

---

Listing 21: Hauptprogramm: kameraseiteUpdate()

## [ def kameraGenerator() ]:

Diese Funktion ist die Hauptfunktion für die Raspberry Pi Kamera. Im Code wird das Kameraproblem erwähnt, bei Unklarheit siehe 5.2.2.5.

```
1 def kameraGenerator():  
2     #Starten der Kamera  
3     kameraAufnahme = cv2.VideoCapture(0)  
4  
5     #Die Kamera soll dauerhaft Bilder aufnehmen.  
6     while True:  
7         #Aufnahme des Frames  
8         ret, frame = kameraAufnahme.read()  
9  
10        #Spiegelung des Frames, um das Kameraproblem zu beheben beziehungsweise zu umgehen.  
11        frameMirror = cv2.rotate(frame, cv2.ROTATE_180)  
12  
13        #Einbinden des MOTIONsensors in das Bild.  
14        frameMirror = cv2.putText(frameMirror, bewegungScannen(), (50,50),cv2  
15                                .FONT_HERSHEY_SIMPLEX, 1,(0,0,0), 1, cv2.LINE_AA, False)  
16  
17        #Komprimieren des Frames und vorläufiges Lagern im Memory Buffer.  
18        ret, frameMirror = cv2.imencode('.jpeg',frameMirror)  
19  
20        #Rückgabe des Generators.  
21        yield (b'--frame\r\n'  
22                b'Content-Type: image/jpeg\r\n\r\n' + frameMirror.tobytes() + b'\r\n')
```

---

Listing 22: Hauptprogramm: kameraGenerator()

Zur Erklärung der Funktionsparametern von cv2.putText(), siehe den Link in der Fußzeile.<sup>34</sup>

---

<sup>34</sup>Quelle: 21.03.2021, <https://bit.ly/3w1pu47> (Link mit bitly.com gekürzt)

---

### [ def liveuebertragung() ]:

Die Funktion garantiert die dauerhafte Liveübertragung über den Webserver.

---

```
1 @app.route('/liveuebertragung')
2 def liveuebertragung():
3     #Rückgabe der Antwort der "kameraGenerator()" Funktion als Frame.
4     return Response(kameraGenerator(),
5                     mimetype='multipart/x-mixed-replace; boundary=frame')
```

---

Listing 23: Hauptprogramm: def liveuebertragung()

---

### 5.3.2.6 Vorbereitung des Fernzugriffs

Damit der Webserver per HTTPS über den Port 443 laufen kann, wird ein eigenes Zertifikat erstellt. Dafür wird im Raspberry Pi Terminal zum Ordner “Diplomarbeit” navigiert und anschließend folgendes durchgeführt.<sup>35</sup>

---

```
1 $ openssl req -x509 -newkey rsa:4096 -nodes -out cert.pem -keyout key.pem -days 365
```

---

Listing 24: Hauptprogramm: Erstellen von cert.pem und key.pem

Nach Ausführung des Befehls wird man über die gewollten Eigenschaften des Zertifikats gefragt. Hier können Attribute wie Name des Zertifikaterstellers, Organisation und weitere Daten angegeben werden. Nachdem dies erledigt ist, sollten sich im Hauptordner “Diplomarbeit” zwei neue Dateien befinden. “cert.pem” und “key.pem”. Diese können nun in “main.py” eingebunden werden. Dies wird folgenderweise implementiert.

---

```
1 if __name__ == "__main__":
2     try:
3         #cert.pem und key.pem werden eingebunden
4         #Als Host wird 0.0.0.0 konfiguriert.
5         #Als Port wird 443 verwendet.
6         app.run(ssl_context=('cert.pem', 'key.pem'), host='0.0.0.0', port=443, debug=False)
7
8     #Ausgeben von exceptions, um Prozesse besser nachvollziehen zu können.
9     except Exception as e:
10        print(e)
```

---

Listing 25: Hauptprogramm: Umsetzung des Fernzugriffs im Hauptprogramm

Das Ergebnis dieser Implementierung kann unter 5.4.3. eingesehen werden.

---

<sup>35</sup>Quelle: 17.03.2021, <https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https>

---

### 5.3.3 Frontend

#### 5.3.3.1 Installation JustGage

Auf den GitHub-Pages von JustGage, kann der Downloadlink gefunden werden.<sup>36</sup>

Anschließend wird das ZIP Verzeichnis heruntergeladen und in /Desktop/Diplomarbeit/static entpackt. Benötigt werden nur folgende zwei Dateien:

- raphael-2.1.4.min.js
- justgage.js

#### 5.3.3.2 Finale Ordnerstruktur

```
> Desktop
    > Diplomarbeit
        > static
            > style.css
            > justgage.js
            > raphael-2.1.4.min.js
        > templates
            > index.html
            > kamera.html
        > Bewegungs_Sensor.py
        > DHT_Sensor.py
        > Datenbank.db
        > main.py
        > cert.pem
        > key.pem
```

Abbildung 28: Hauptprogramm: Finale Ordnerstruktur

---

<sup>36</sup>Quelle: 17.03.2021, <https://toorshia.github.io/justgage/>

---

### 5.3.3.3 [ index.html ]

index.html stellt die Hauptwebseite dar, beziehungsweise die "Landing-Page".

#### [ header ]

Die Implementierung des Headers, sieht wie folgt aus.

---

```
1 <head>
2     <!-- Webseitentitel -->
3     <title>Diplomarbeit: Home Security Prototyp</title>
4
5     <!-- Einbinden von Style.css -->
6     <link rel="stylesheet" href='../static/style.css' />
7
8     <!-- Einbinden von Bootstrap -->
9     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
10         rel="stylesheet" integrity="sha384-giJF6kkoqNQ0Ovy+HMDP7az0uL0xtbfIcaT9wjKHR8RbDVddVHyTfAAAsrekwKmP1"
11         crossorigin="anonymous">
12
13     <!-- Definieren von Content-Type -->
14     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
15
16
17     <style>
18         @import url('https://fonts.googleapis.com/css2?family=Yusei+Magic&display=swap');
19         body {
20             text-align: center;
21             background-color: rgb(24, 24, 24);
22             color: white
23         }
24
25         /* Konfiguration Gages 1 und 2 */
26         #g1,
27         #g2 {
28             width: 500px;
29             height: 460px;
30             display: block;
31         }
32
33         /* Konfiguration CSS Cards */
34         .cards {
35             border: black;
36             background-color: rgb(24, 24, 24);
37             width: 30rem;
38         }
39     </style>
40 </head>
```

---

Listing 26: Hauptprogramm: index.html - header

---

## [ body teil 1 ]

Der erste Teil des HTML-Bodys kümmert sich weitestgehend um die Temperatur- und Luftfeuchtigkeitsdiagramme.

---

```
1 <body>
2     <!-- Header bzw. Diplomarbeits-Titel -->
3     <h1>Home Security Prototyp </h1>
4
5     <div class="container">
6         <div class="row">
7             <div class="col-lg-6">
8                 <div id="g1">
9                     </div>
10                <div id="g2" class="mb-10">
11                    </div>
12            </div>
13
14     <!-- col-lg ist eine Bootstrap Klasse zum Bau von Layouts mittels Grid-Systemen -->
15     <div class="col-lg-6">
16         <div class="card cards text-center mt-3 mx-auto" style="width: 100%;">
17             <div class="card-body">
18                 <!-- Einbinden des Temperaturdiagrammes -->
19                 
21             </div>
22         </div>
23         <div class="card cards text-center mt-3 mx-auto" style="width: 100%;">
24             <div class="card-body">
25                 <!-- Einbinden des Luftfeuchtigkeitsdiagrammes -->
26                 
28             </div>
29         </div>
30     </div>
31     </div>
32 </div>
```

---

Listing 27: Hauptprogramm: index.html - body teil 1

---

## [ body teil 2 ]

Im zweiten Teil werden die Bootstrap-Buttons implementiert.

```
1   <div class="container navigation">
2     <div class="row">
3       <div class="container col-lg-6">
4         <div class="row">
5           <div>
6
7             <!-- Erstellen des Buttons, welcher die Zeit der letzten Messung anzeigt -->
8             <div class="card cards text-center mx-auto">
9               <div class="card-body">
10                 <h4 class="card-title">Zuletzt gemessen: {{ zeit }}</h4>
11                 <a href="/" class="btn button btn-primary">aktualisieren</a>
12               </div>
13             </div>
14           </div>
15
16             <!-- Erstellen des Buttons, welcher auf die Kameraseite navigiert -->
17             <div class="card cards text-center mt-3 mx-auto">
18               <div class="card-body">
19                 <h4 class="card-title">Livestream</h4>
20                 <a href=".//kamera" class="btn button btn-primary">navigieren </a>
21               </div>
22             </div>
23           </div>
24         </div>
25
26         <div class="col-lg-6">
27
28           <!-- Erstellen des Buttons, durch welchen die Datenbank ausgewertet werden kann -->
29           <div class="card cards text-center mt-3 mx-auto">
30             <div class="card-body">
31               <h4 class="card-title">Datenbank auswerten</h4>
32               <p> Betrachtungsintervall (Ein Schritt entspricht: {{ abtastung }} Minute)
33                 <form method="POST">
34                   <input name="anzahlGeforderterWerte" value={{anzahlGeforderterWerte}}>
35                   <input type="submit" class="btn button btn-primary">
36                 </form>
37               </p>
38             </div>
39           </div>
40         </div>
41       </div>
42     </div>
```

---

Listing 28: Hauptprogramm: index.html - body teil 2

---

## [ body teil 3 ]

Im dritten Teil werden die JustGage Elemente implementiert.

```
1  <!-- Einbinden der JustGage Plugins. -->
2  <script src="../static/raphael-2.1.4.min.js"></script>
3  <script src="../static/justgage.js"></script>
4  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"
5         integrity="sha384-q2kxQ16AaE6UbzuKqyBE9/u/KzioAlnx2maXQHiDX9d4/zp80k3f+M7DPm+Ib6IU"
6         crossorigin="anonymous"></script>
7  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.min.js"
8         integrity="sha384-pQQkAEnwaBkjppZ8RU1fF1AKtTcHJwF13pb1pTlHXybJjHpMYo79HY3hIi4NKxyj"
9         crossorigin="anonymous"></script>
10
11 <!-- Erstellen des Skriptes für die JustGage Implementierung -->
12 <script>
13     var g1, g2;
14     document.addEventListener("DOMContentLoaded", function(event) {
15
16         /* Implementieren des Temperatur-Gages */
17         g1 = new JustGage({
18             id: "g1",
19             value: {{temperatur}},
20             valueFontColor: "white",
21             titleFontColor: "white",
22             min: -10,
23             max: 50,
24             gaugeColor: "Dodgerblue",
25             title: "Temperatur",
26             label: "Celcius"
27         });
28
29         /* Implementieren des Luftfeuchtigkeits-Gages */
30         g2 = new JustGage({
31             id: "g2",
32             value: {{luftfeuchtigkeit}},
33             valueFontColor: "white",
34             titleFontColor: "white",
35             gaugeColor: "Dodgerblue",
36             min: 0,
37             max: 100,
38             title: "Luftfeuchtigkeit",
39             label: "%"
40         });
41     });
42     </script>
43 </body>
```

---

Listing 29: Hauptprogramm: index.html - body teil 3

---

#### 5.3.3.4 [ kamera.html ]

Die kamera.html stellt die Liveübertragung auf einer Subdomain dar.

##### [ head ]

Im dritten Teil werden die JustGage Elemente implementiert.

---

```
1 <html>
2 <head>
3     <!-- Webseitentitel -->
4     <title>Live Videoüberwachung</title>
5
6     <!-- Einbinden von Bootstrap -->
7     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
8         rel="stylesheet" integrity="sha384-giJF6kkoqNQ00vy+HMDP7az0L0xtbfIcaT9wjKHR8RbDVddVHyTfAAsrekwKmP1"
9         crossorigin="anonymous">
10    <link rel="stylesheet" href='./static/style.css' />
11
12    <!-- Style import -->
13    <style>
14        @import url('https://fonts.googleapis.com/css2?family=Yusei+Magic&display=swap');
15        body {
16            text-align: center;
17            background-color: rgb(24, 24, 24);
18            color: white
19        }
20    </style>
21 </head>
```

---

Listing 30: Hauptprogramm: kamera.html - head

---

## [ body ]

Im dritten Teil werden die JustGage Elemente implementiert.

```
1  <body>
2      <div class="container">
3          <div class="row">
4              <!-- Bootstrap-Klasse -->
5              <div class="col-lg-12">
6                  <!-- Seitentitel -->
7                  <h1>Live Videoüberwachung</h1>
8                  <!-- Einbinden der OpenCV Kameraimplementation in die Webseite -->
9                  <h3></h3>
10             </div>
11         </div>
12     </div>
13     <div class="mx-auto return">
14         <!-- Hardcoding des Standorts in Form eines Textes -->
15         <p style="margin-top: 40px">Location: Innsbruck, Austria</p>
16     </div>
17
18     <!-- Einbinden von Plugins und Libraries -->
19     <script src="../static/raphael-2.1.4.min.js"></script>
20     <script src="../static/justgage.js"></script>
21     <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"
22            integrity="sha384-q2kxQ16AaE6UbzuKqyBE9/u/KzioAlnx2maXQHiDX9d4/zp80k3f+M7DPm+Ib6IU"
23            crossorigin="anonymous"></script>
24     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.min.js"
25            integrity="sha384-pQQkAEnwaBkjpqZ8RU1fF1AKtTcHJwF13pb1pT1HXybJjHpMYo79HY3hIi4NKxyj"
26            crossorigin="anonymous"></script>
27
28 </body>
29 </html>
```

---

Listing 31: Hauptprogramm: kamera.html - body

---

### 5.3.3.5 [ style.css ]

Mit style.css wird das Webseiten-Layout kontrolliert. Die Implementierung sieht wie folgt aus.

```
1  /* Styling des Bodies */
2  body {
3      padding: 1%;
4      text-align: center;
5      overflow-x: hidden;
6  }
7
8  /* Styling des Header 1 */
9  h1 {
10     font-family: 'Yusei Magic', sans-serif;
11 }
12
13 /* Styling des buttons */
14 .button {
15     width: 10rem;
16 }
17
18 /* Anpassung der navigation */
19 .navigation {
20     margin-top: 40px;
21 }
22
23 /* Styling des returns */
24 .return {
25     left: 50%;
26     top: 50%;
27 }
28
29 /* Styling des containers */
30 .container {
31     position: relative;
32     width: 100%;
33 }
34
35 /* Styling der JustGage - Gages */
36 #g1,
37 #g2 {
38     margin: 20px;
39     display: inline-block;
40     width: 200px;
41     height: 160px;
42 }
```

---

Listing 32: Hauptprogramm: style.css

---

## 5.4 Fernzugriff

### 5.4.1 Programmintern

Im Hauptprogramm wurde der Fernzugriff schon vorbereitet. Siehe dafür 5.3.2.6.

### 5.4.2 Programmextern

Für den Fernzugriff muss Portweiterleitung beziehungsweise Port-Forwarding in den Router-Einstellungen vorgenommen werden. Der verwendete Router ist die UPC-Connect-Box. Da dieser Router standardmäßig keine Port-Forwarding-Funktion bereitstellt, wurde hierfür der Internetprovider (Magenta) verständigt. Nach einigen Werktagen wurde dieses Feature freigeschaltet. Nun konnte Port-Forwarding konfiguriert werden. Dabei wird die Adresse 192.168.0.108 (IPv4-Adresse des Raspberry Pi) über Port 443 mit UDP- und TCP-Protokollen konfiguriert. Das Ergebnis sieht anschließend wie folgt aus:

The screenshot shows the 'Portweiterleitung' (Port Forwarding) configuration page. On the left, there's a sidebar with icons for 'Startseite', 'Verbundene Geräte', 'Bridge Mode', and 'Erweiterte Einstellungen'. The main area has a green success message: 'Ihre Einstellungen wurden übernommen.' Below it, a table lists port forwarding rules:

Lokal	Extern				
IP Adresse	Port Range	Port Range	Protokoll	Aktiviert	Löschen
192.168.0.108	443	443	Both	<input checked="" type="checkbox"/>	<input type="checkbox"/>
192.168.0.150	33160	33160	UDP	Automatisch hinzugefügt durch UPnP	

Abbildung 29: Fernzugriff: Portweiterleitung

### 5.4.3 Zertifikatsergebnis

Das Web-Zertifikat, welches konfiguriert wurde (siehe 5.3.2.6.) sieht demnach wie folgt aus.

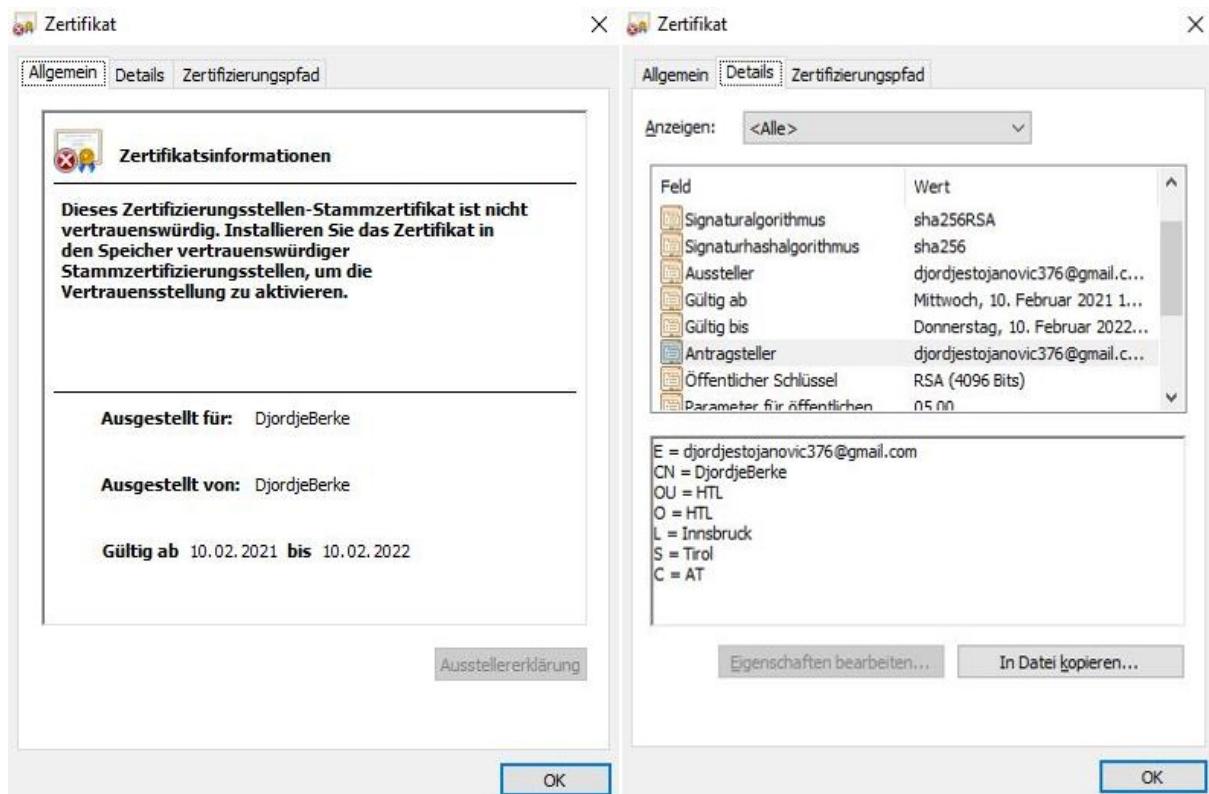


Abbildung 30: Fernzugriff: Web-Zertifikat im Detail

Das Zertifikat läuft genau ein Jahr lang, wie in 5.3.2.6 mit folgendem Befehl konfiguriert wurde.

```
$ openssl req -x509 -newkey rsa:4096 -nodes -out cert.pem -keyout key.pem -days 365
```

---

## 5.5 Gehäuse: Entwicklung, Produktion und Nachbearbeitung

Es gibt unzählige Möglichkeiten für so ein System ein Gehäuse zu entwickeln. Man kann viele Vorlagen auf diversen Online-Shops wie zum Beispiel "Amazon.com" erwerben. Da in diesem Projekt das Ziel war eine 100 prozentige Individualität anzustreben, wurde ein Gehäuse von 0 auf selbst konzipiert und produziert. Das Gehäuse in diesem Projekt wurde selbstdesignt und 3D-gedruckt.

### 5.5.1 Selbstgestellte Bedingungen bezogen auf Funktionalität

Ein Gehäuse soll...

- ...das **Innere** schützen,  
⇒ Um die Komponenten, die am Raspberry Pi verbaut wurden, vor Umwelteinflüssen zu schützen, wurde auf eine starke Gehäusewand wertgelegt. Die Stärke beziehungsweise Dicke jeder Seite sollte demnach mindestens 0.5cm betragen. Das Gehäuse wird somit sehr robust und absorbiert auch gleichzeitig Stöße, da das Gehäuse aus einem thermoplastischen Kunststoff bestehen soll und daraus folgend auch eine gewisse Elastizität aufweist.
- ...eine **tarnende Funktion** mit sich bringen  
⇒ Da es sich hiermit um ein Überwachungssystem handelt, dürfen von außen auch keine Leuchtsignale beziehungsweise LEDs erkennbar gemacht werden, da diese den Standort des Geräts direkt verraten würden. Auch die Farbe wurde bedacht. Hier wurde ein schwarzes Material gewählt, da es lichtabsorbierend wirkt und es demnach, besonders in der Dunkelheit, schwer zu sichten ist.
- ...und **optisch ansprechend** sein.  
⇒ Es wurde Wert auf Unkompliziertheit gelegt. Daher wurde ein einfaches, rechteckiges Gehäuse mit nur den benötigten Aussparungen entwickelt. Die schlichte Farbe ist auch eine Bereicherung der Optik.

---

### 5.5.2 Informationen zum Druck



Abbildung 31: Stratatys Fortus 900 MC HighTech-Industriedrucker

Das 2D-Modell wurde von AutoCad auf Catia in ein 3D-Modell umdesignt - natürlich würde eine 3D-Zeichnung auf AutoCad genauso gut funktionieren. Dieses Design wurde danach als .stl Datei abgespeichert beziehungsweise geplottet. Die .stl Datei wurde zudem mit Stratasys Insight 12.6 aufbereitet. Dabei wurde auf die richtige Schichtstärke, auf das richtige Material und auf das richtige Supportmaterial geachtet. Nun wurde ein „Druckjob“ erstellt und über Stratasys Control Center 12.6 an die Maschine übergeben. Die Blackbox selber wurde mit Stratasys Forus 900 MC 3D-gedruckt. Als Druckmaterial wurde ABS M30 (in schwarz) verwendet und es wurden genau 172 cm<sup>3</sup> davon verwendet. Das Supportmaterial SR30 wurde für den Ausschnitt (USB+LAN) benutzt. Hierbei wurden 21 cm<sup>3</sup> verwendet. Als Düse wurde eine T16 Düse mit einer Schichtstärke von 0.25mm benutzt. Die Druckdauer betrug ziemlich genau vier Stunden. Zu beachten ist, dass zwei Blackboxen gedruckt wurden. Eine Blackbox fiel den Experimenten und den Belastungstests beziehungsweise Performancetests zum Opfer. Der 3D-Druck wurde mithilfe der Firma INNIO Jenbacher GmbH durchgeführt. Die Kosten

---

wären für eine Privatperson mit dieser HighTech-Ausrüstung unnötig hoch, da auch ABS benutzt wurde. Es ist ein relativ teures Material und darausfolgend ein teurer Druck. Der circa 190 cm<sup>3</sup> Materialverbrauch kann man hiermit auf ungefähr 30€ ansiedeln - pro Blackbox 15€. Schließlich kann man für so einen einfachen Gehäusedruck auch andere billigere, unpräzisere 3D-Drucker nutzen.

### 5.5.3 Skizzen und Ideen

Vor jeder Gehäuseentwicklung müssen Skizzen und Ideen vom Kopf aufs Papier gebracht werden. Genau 22 Blackbox-Skizzen wurden gezeichnet und diskutiert. Die Auswahl fiel auf eine kompakte, schlichte, rechteckige Bauweise - somit hat auch der 3D-Druck selbst einen relativ kleinen Zeitaufwand. Da es unnötig wäre alle 22 Skizzen in dieser Dokumentation einzubinden, haben wir die drei Skizzen ausgewählt, die in die engere Auswahl kamen.

#### 5.5.3.1 Engere Auswahl

##### Skizze Nr. 1

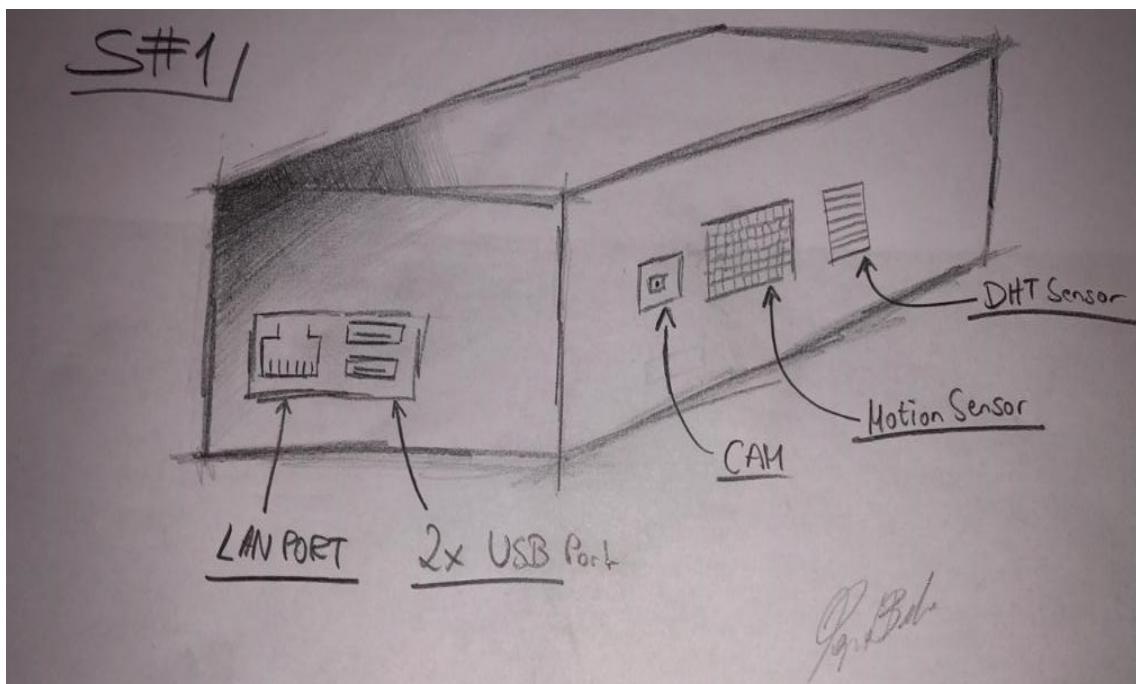


Abbildung 32: Gehäusebau: Skizze 1

##### Vorteile:

- horizontal aufstellbar
- kein Problem mit Power Port (gegenüber von den Komponenten)

##### Nachteile:

- Komplikationen im Innenraum bezüglich Platz
- Höhe muss berücksichtigt werden

---

## Skizze Nr. 2

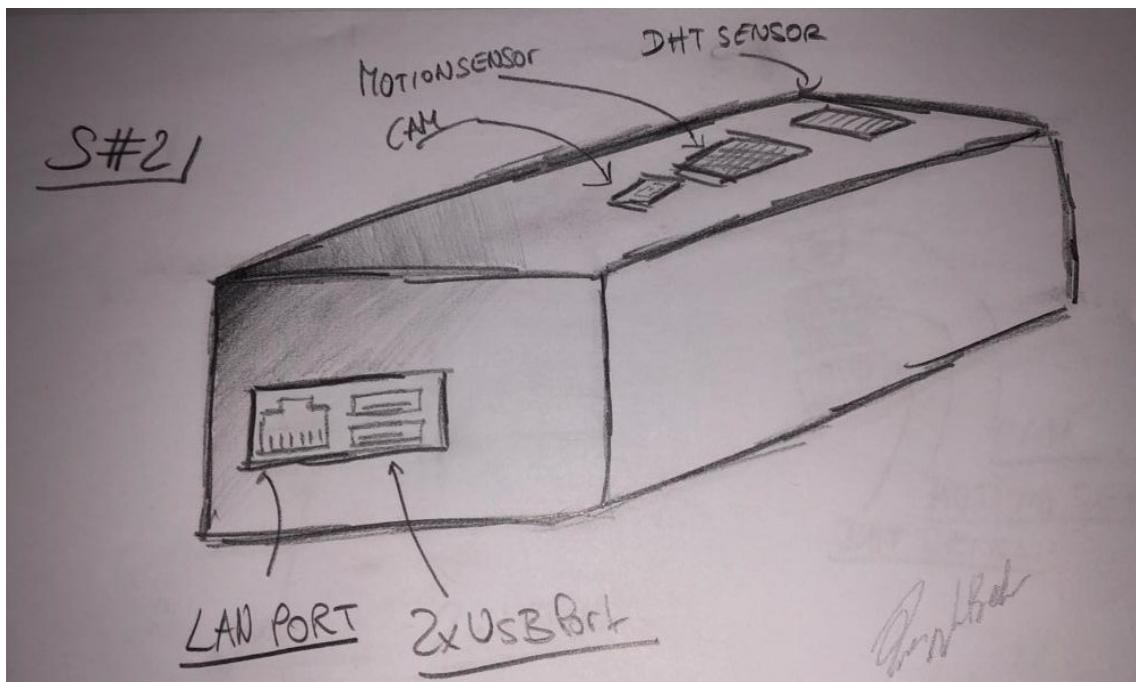


Abbildung 33: Gehäusebau: Skizze 2

### Vorteile:

- optisch ansprechender
- Höhe ist irrelevant, da sich die Komponenten oben befinden

### Nachteile:

- muss quer aufgestellt werden (Kamera schaut sonst nach oben)
- Power Port ist oben (wenn es quer aufgestellt wurde)

---

### Skizze Nr. 3

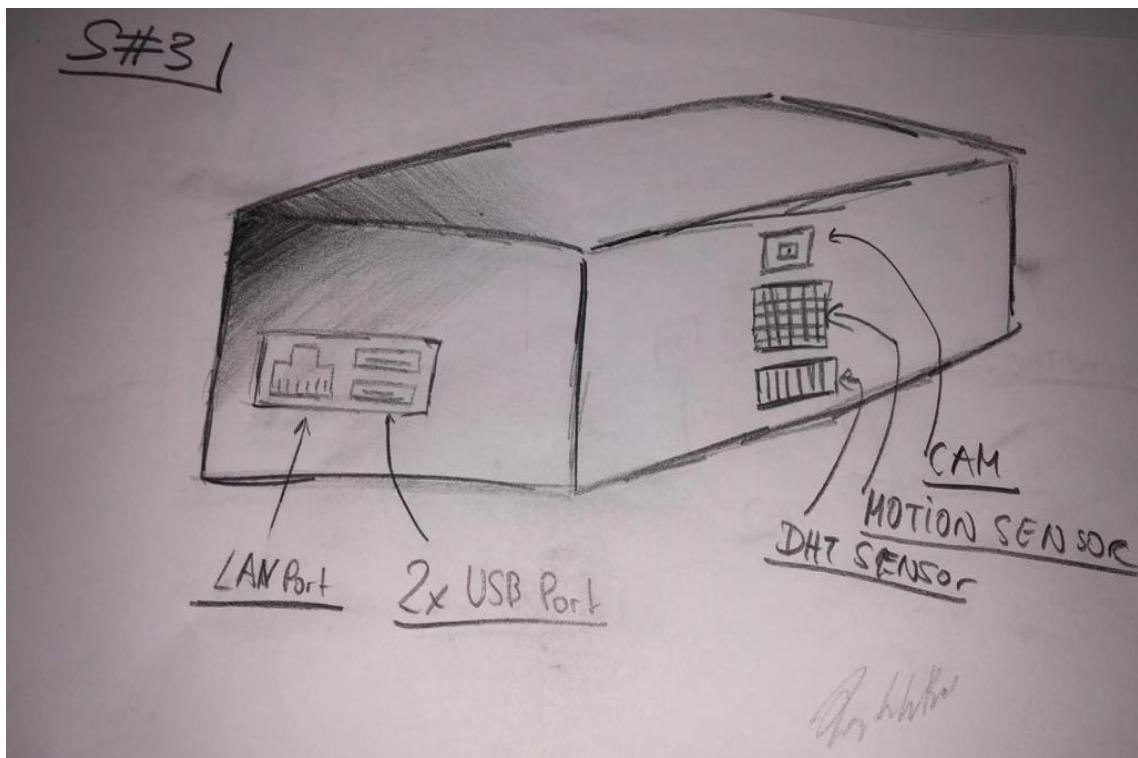


Abbildung 34: Gehäusebau: Skizze 3

#### Vorteile:

- kann horizontal aufgestellt werden
- die Länge des Gehäuses kann so lang wie der Raspberry Pi selbst sein

#### Nachteile:

- Komplikationen im Innenraum bezüglich Platz
- es muss dementsprechend hoch gebaut werden, da „Staplung“ der Komponenten

#### 5.5.3.2 Entgültige Auswahl

Am Ende wurde Skizze 2 (eigentlich: Skizze 8 von 22) ausgewählt, da es wichtig war, dass das Gehäuse nicht so hoch gebaut wird. Ein Nachteil ist es, dass man nun das Gerät quer stellen muss, damit die Kamera nicht nach oben, sondern in die gewünschte Richtung (horizontal) schaut. Dies stellt jedoch keinen Performanceverlust dar. Auch die Handlichkeit geht nicht verloren.

#### 5.5.4 Raspberry Pi 3B Plus Dimensionen

Auschnitt aus dem Datenblatt<sup>37</sup>

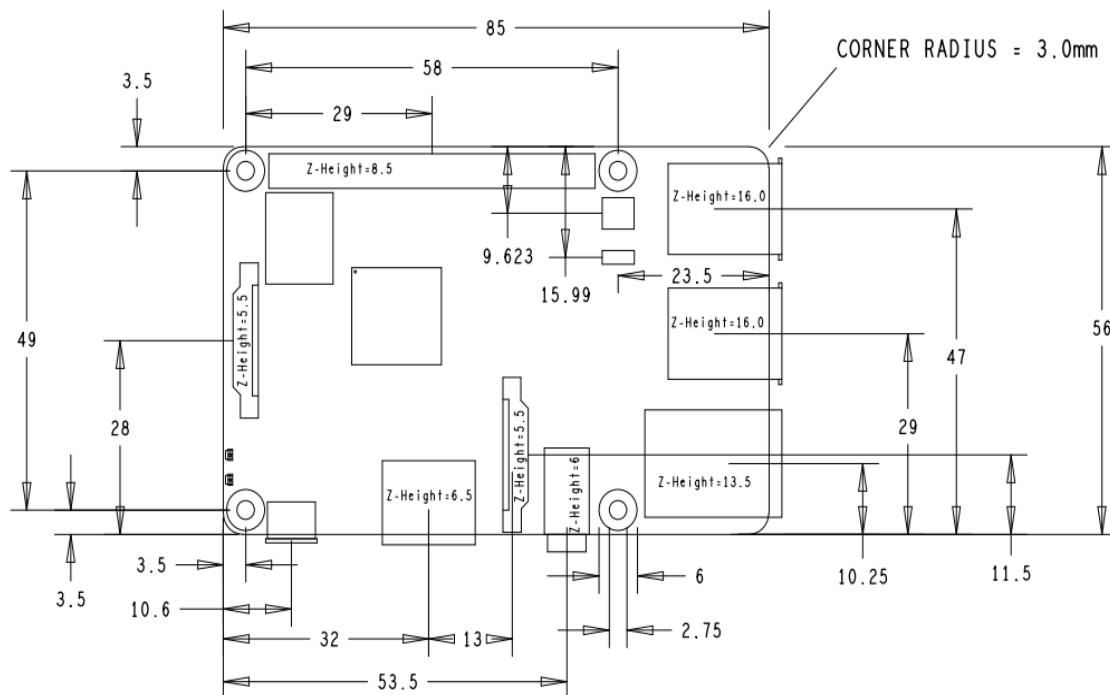


Abbildung 35: Raspberry Pi: Dimensionen

Das Gehäuse wurde so entwickelt, um ausreichend Platz für Kabel und sonstige Einbauten wie Vorwiderstände zu garantieren. Es wurde hier auch auf die Wärmeentwicklung geachtet. Hier wurde eine „illegal“ Zone definiert. Diese befindet sich direkt über den USB-Ports beziehungsweise direkt über dem LAN-Port (rechts im Bild). In dieser Zone wurde nichts verbaut, um die Höhe niedrig zu halten. Es wurden demnach Ausschnitte für 1 x LAN und 2 x USB mitberücksichtigt - für die Funktion ist nur ein LAN-Port vonnöten. Die 2 x USB Ausschnitte sind dafür gedacht, falls man für den Bootvorgang beziehungsweise Hauptprogrammstart Tastatur und Maus benötigen sollte. Die Löcher für die Befestigung des Raspberry Pi wurde schon bei der CAD-Zeichnung beziehungsweise beim 3D-Design mitberücksichtigt. Die Löcher für die Komponentenbefestigung und auch die Aussparung für den Power Port wurden in der Werkstatt nachbearbeitet. Es wurde versucht die Basis so kompakt wie möglich zu halten.

<sup>37</sup>Quelle: 13.03.21, <https://bit.ly/3roYmZk> (Link mit bitly.com gekürzt)

---

## 5.5.5 Entwicklung und Design der Blackbox

### 5.5.5.1 Design in AutoCad

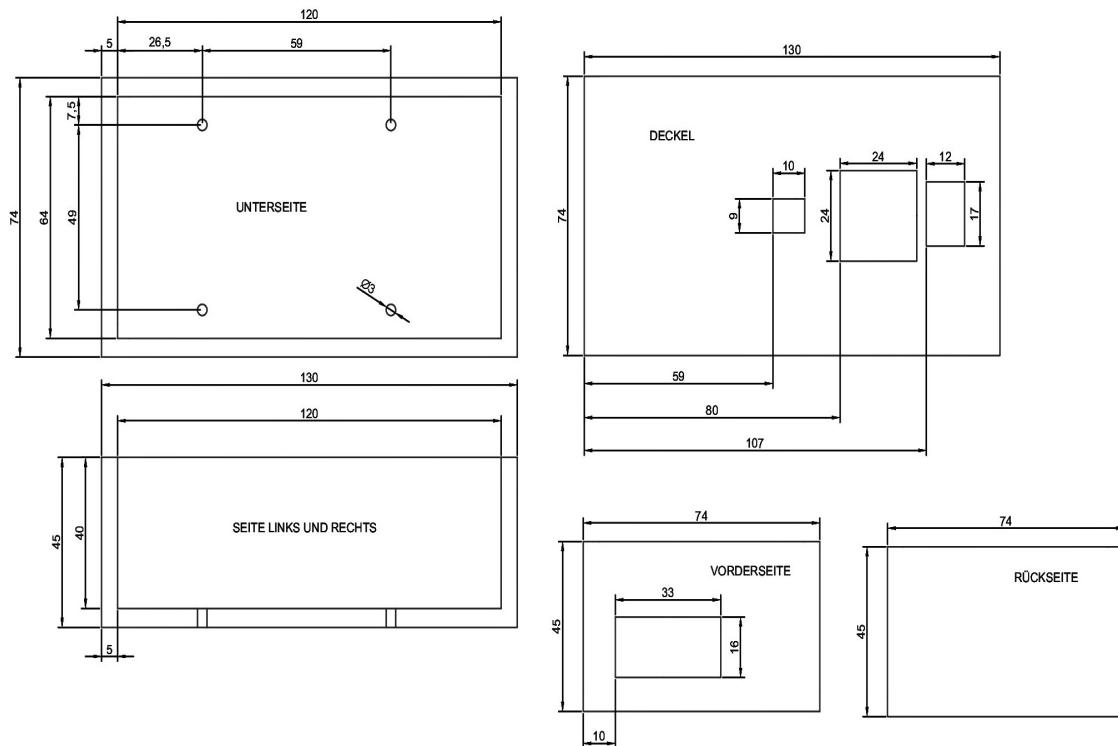


Abbildung 36: AutoCad-Plot: Gesamtplot

Um jede Seite gleichzeitig zu sehen, um Sicherzugehen, ob der Gedankengang funktioniert und es sich so ausgehen würde wie gedacht, wurde extra eine AutoCad-Zeichnung entworfen. Hier wurde die 0.5cm Wandstärke mitberücksichtigt. Alle Überlegungen bezüglich der Längen und Breiten werden in den folgenden Seiten näher erklärt.

---

#### 5.5.5.1.1 AutoCad: Unterseite

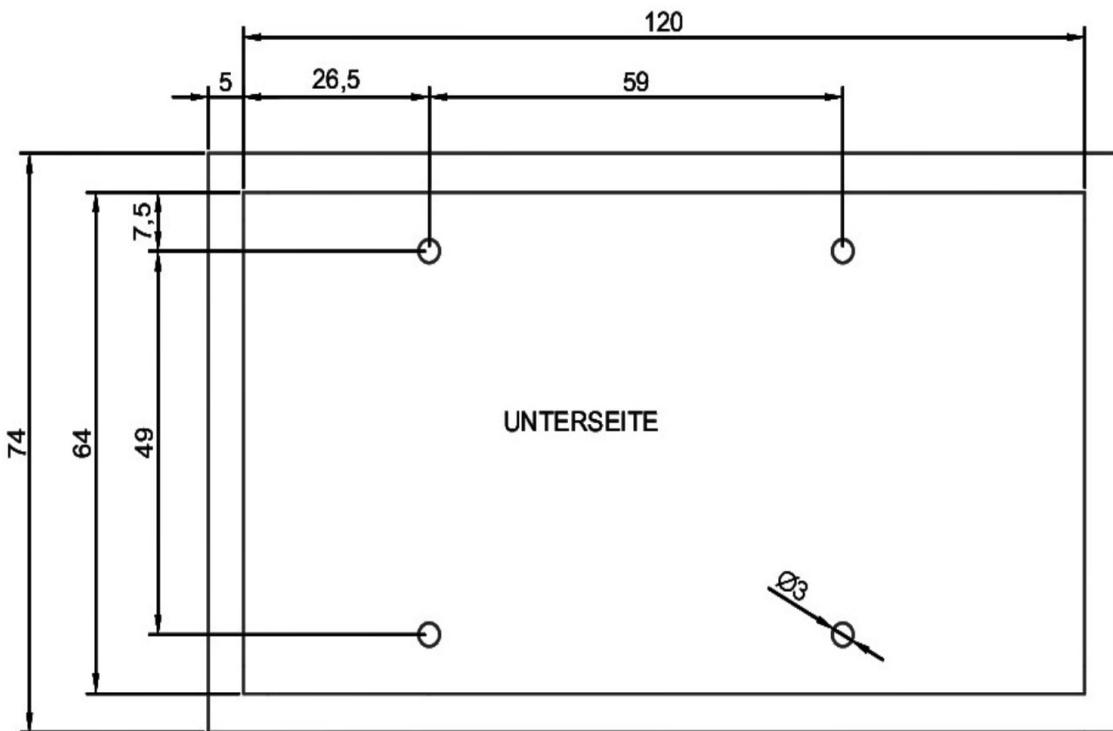


Abbildung 37: AutoCad-Plot: Unterseite

- Tiefe: 120mm  $\Rightarrow$  RPi Standardlänge + 35mm (exklusive 5mm Wandstärke)
- Breite: 74mm  $\Rightarrow$  RPi Standardbreite + 18mm (inklusive 5mm Wandstärke)
- Befestigung: 3mm  $\Rightarrow$  RPi Standarddurchmesser + 0.25mm

---

#### 5.5.5.1.2 AutoCad: Deckel

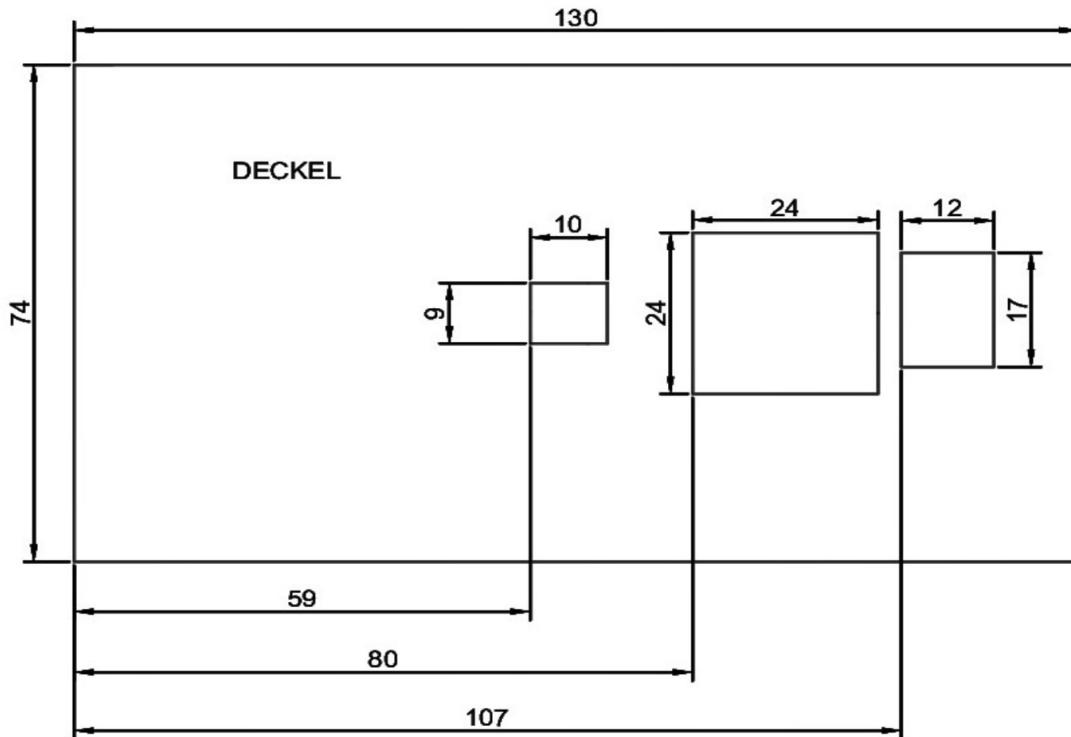


Abbildung 38: AutoCad-Plot: Deckel

- Tiefe: 130mm  $\Rightarrow$  RPi Standardlänge + 45mm (inklusive 5mm Wandstärke)
- Breite: 74mm  $\Rightarrow$  RPi Standardbreite + 18mm (inklusive 5mm Wandstärke)
- Kamera (links): 10x9mm
- Motion Sensor (mittig): quadratisch 24mm
- Temperatur- und Luftfeuchtigkeitssensor (rechts): 12x17mm

---

#### 5.5.5.1.3 AutoCad: Seiten

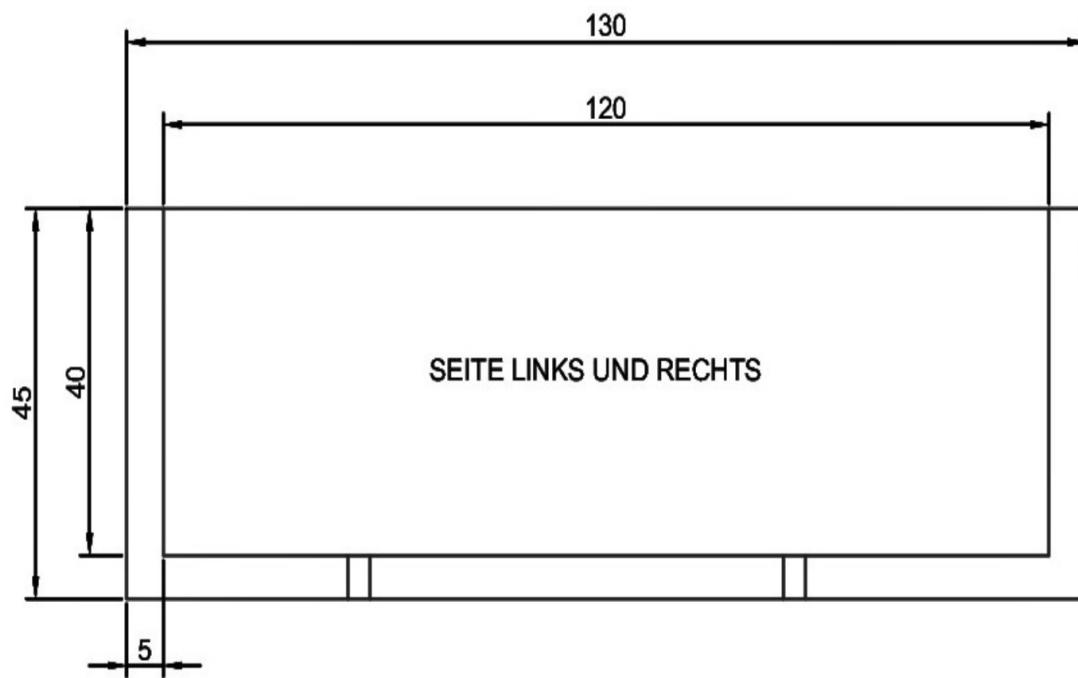


Abbildung 39: AutoCad-Plot: Seiten (Links und Rechts)

- Tiefe: 130mm  $\Rightarrow$  RPi Standardlänge + 45mm (inklusive 5mm Wandstärke)
- Höhe: 45mm  $\Rightarrow$  RPi Standardbreite + 27mm (inklusive 5mm Wandstärke)

---

#### 5.5.5.1.4 AutoCad: Vorderseite

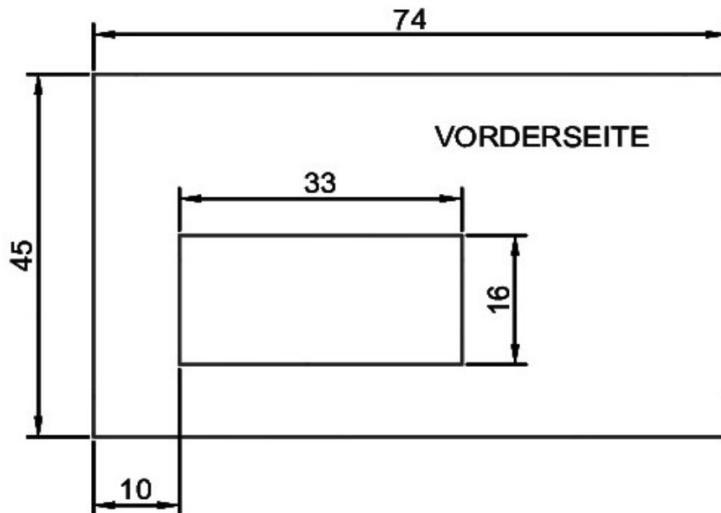


Abbildung 40: AutoCad-Plot: Vorderseite

- Breite: 74mm  $\Rightarrow$  RPi Standardlänge + 18mm (inklusive 5mm Wandstärke)
- Höhe: 45mm  $\Rightarrow$  RPi Standardbreite + 27mm (inklusive 5mm Wandstärke)
- 1 x LAN-Port + 2 x USB-Port: 33 x 16 mm

---

#### 5.5.5.1.5 AutoCad: Rückseite

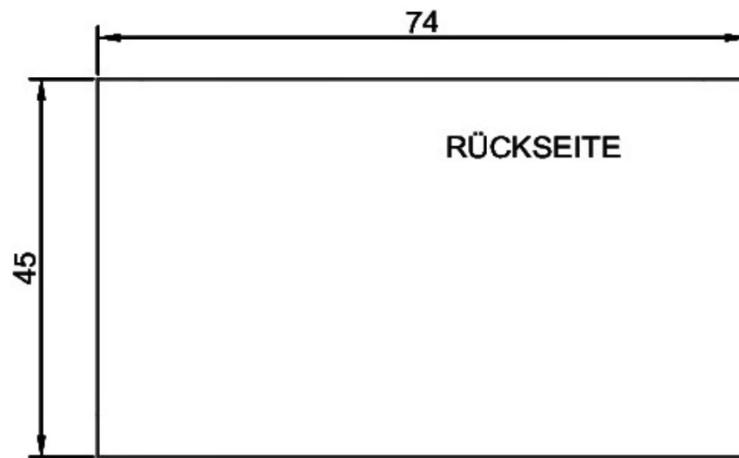


Abbildung 41: AutoCad-Plot: Rückseite

- Breite: 74mm  $\Rightarrow$  RPi Standardlänge + 18mm (inklusive 5mm Wandstärke)
- Höhe: 45mm  $\Rightarrow$  RPi Standardbreite + 27mm (inklusive 5mm Wandstärke)

---

### 5.5.5.2 Design in Catia

Nach der Gehäuseentwicklung, Detailarbeit und dem Feinschliff mit AutoCad, wurden diese fünf AutoCad Dateien in zwei (Deckel + Basis) 3D-Druck fähige STL-Dateien (Catia) umdesignt.

#### 5.5.5.2.1 Catia: Deckel

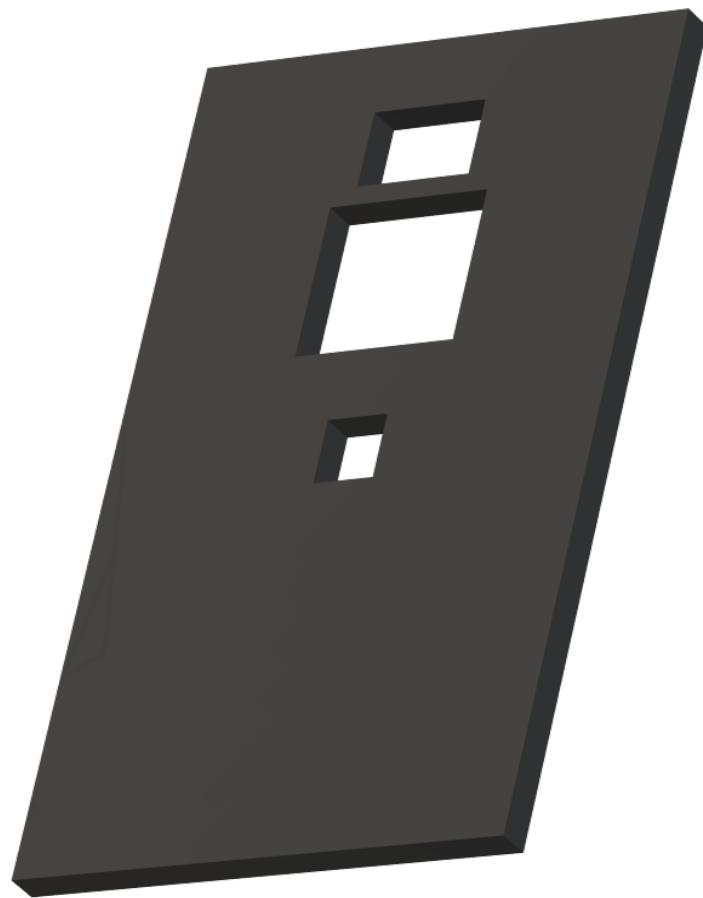


Abbildung 42: STL-Design: Deckel

Hier sieht man die Aussparungen vom Deckel (Kamera, DHT-Sensor und PIR-Sensor). Erkennbar ist auch die Stärke des Deckels (5mm)

---

#### 5.5.5.2.2 Catia: Basis

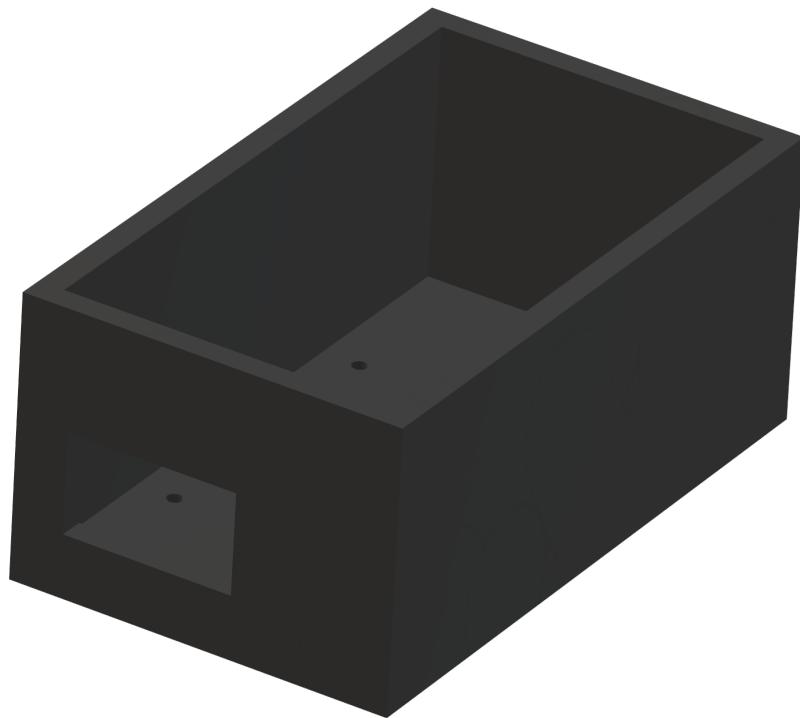


Abbildung 43: STL-Design: Basis

Hier sieht man die Aussparungen vom LAN-Port und den zwei USB-Ports. Erkennbar ist auch die Stärke der Unterseite und der restlichen Seiten (5mm). Wie man sieht ist hier keine Aussparung für den Power Port mitberücksichtigt - dies wird, unter anderem, bei Punkt 5.5.7 näher beschrieben.

---

### 5.5.6 3D-Druck

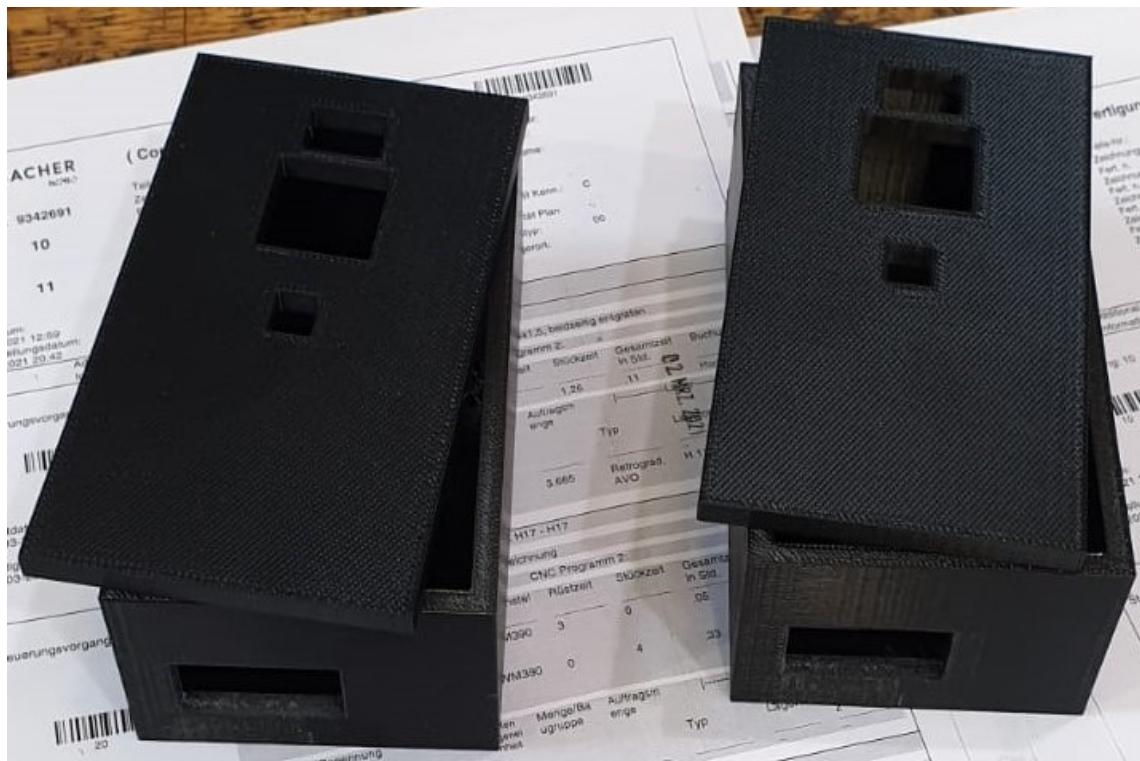


Abbildung 44: 3D-gedrucktes Gehäuse

Hier sieht man das unbearbeitete, fertig gedruckte Gehäuse. Es besteht aus ABS-Kunststoff (Acrylnitril-Butadien-Styrol) und ist daher extrem widerstandsfähig, robust und stoßfest. Die nötigen Bohrungen für die Komponentenbefestigung und die Aussparung für den Power Port wurde in der Werkstatt nachbearbeitet.

---

## 5.5.7 Nachbearbeitung des Gehäuses

### 5.5.7.1 Bohrungen

Die Komponentenbefestigung wurde hiermit sichergestellt. Die Bohrungen wurden nicht vorher geplant, sondern individuell zur Komponente angepasst. Alle Bohrungen bezüglich Komponentenbefestigung (außer Raspberry Pi) wurden nachbearbeitet und haben einen Bohrdurchmesser von 2mm.

Die Deckelbefestigung wurde demnach auch nachbearbeitet. Zuerst wurden in die Eckpunkte der Basis (siehe 5.5.5.2.2) vier 2.5mm Löcher gebohrt. Anschließend wurden diese vier Löcher mit einem M3 Gewindebohrer nachgebohrt. Schließlich wurden auch, passend zur Basis, vier Bohrungen am Deckel vorgenommen. Der Bohrdurchmesser beträgt hierbei 3mm. Nun kann der Deckel (siehe 5.5.5.2.1) mit der Basis (vier Schrauben) verbunden werden.

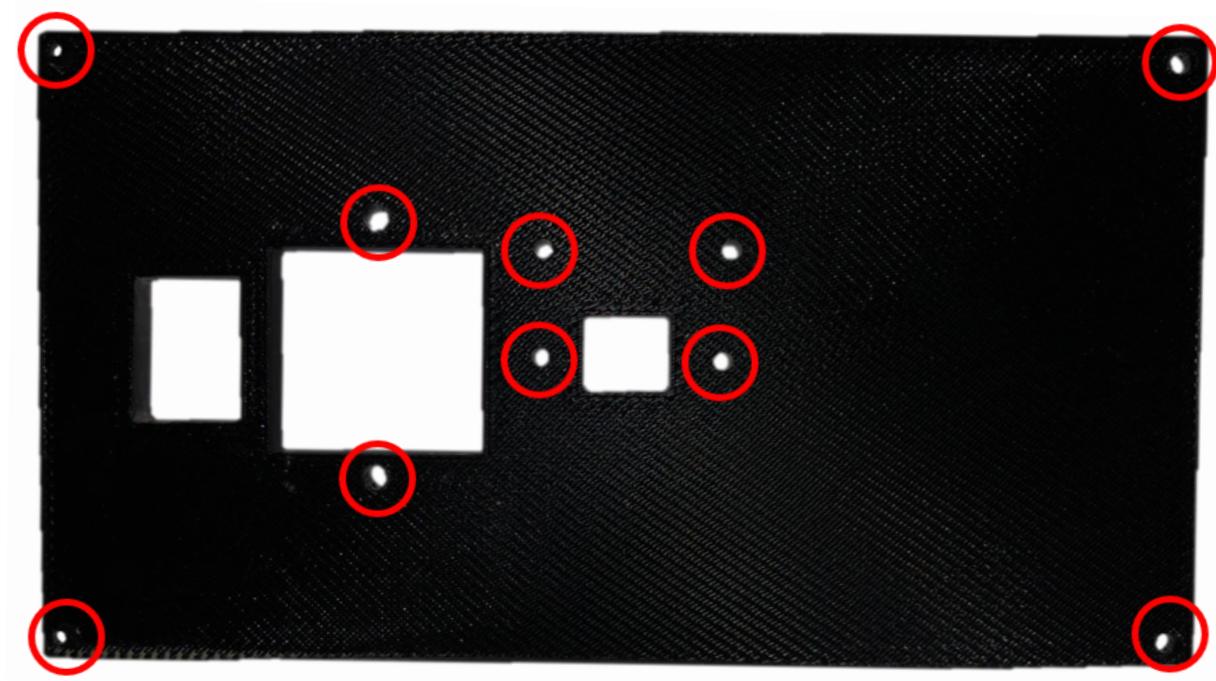


Abbildung 45: Nachbearbeitung: Bohrungen

---

#### 5.5.7.2 Power-Port

Die Bearbeitung des Power-Ports (USB MICRO) wurde demnach auch nachkonzipiert. Hierbei wurde die Platine des Raspberry Pi in das Gehäuse gelegt und fest verschraubt. Nun wurde die genaue Lokation des Power-Ports markiert. Schließlich wurden drei Bohrungen durchgeführt. Daraus wurde ein Viereck (passend zum Port) nachgefeilt. Es wurde darauf geachtet, dass man ohne Probleme das Kabel ein- und ausstecken kann.

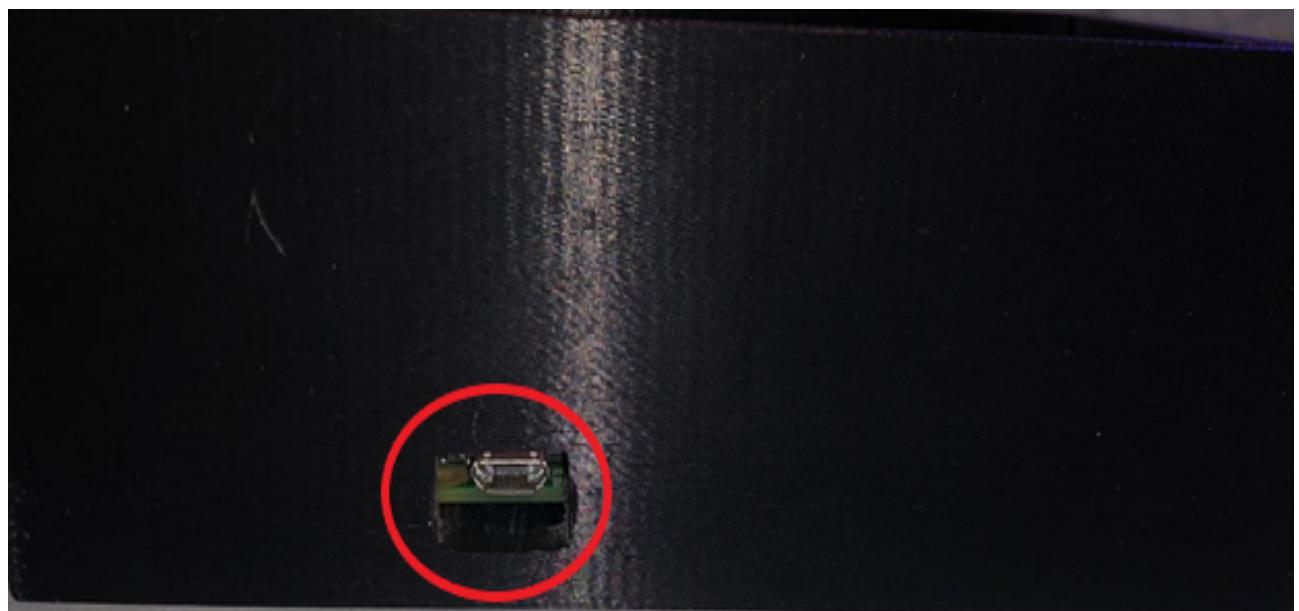


Abbildung 46: Nachbearbeitung: Power-Port

## 5.6 Projektschematik

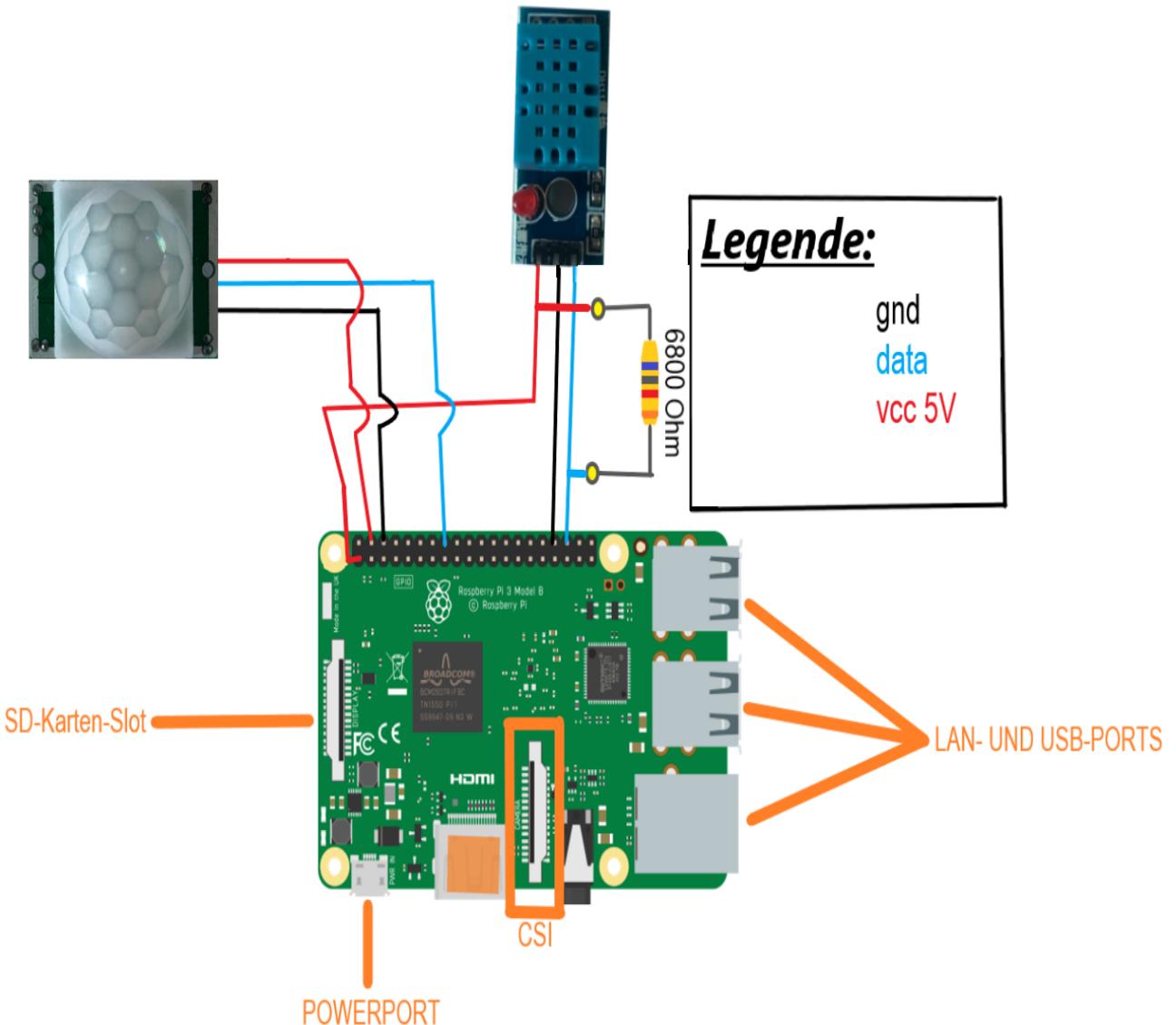


Abbildung 47: Gesamtschaltung ohne Pi Cam, 25.03.2021

Diese Raspberry Pi Abbildung (Abbildung 47) stammt von Wikipedia<sup>38</sup>. Die Bauteile und sonstige Verbindungen wurde extra gezeichnet. Es ist ausdrücklich erlaubt diese Abbildung zu verwenden - siehe<sup>38</sup>.

**Wikipedia-Licensing -**<sup>39</sup>

You are free to share and to remix under following condition: attribution or share alike

Author: Butix, based on works by Lucasbosch and Cymkey, 18.10.16

<sup>38</sup>Quelle: 25.03.2021, [https://upload.wikimedia.org/wikipedia/commons/a/a3/Raspberry\\_Pi\\_3\\_illustration.svg](https://upload.wikimedia.org/wikipedia/commons/a/a3/Raspberry_Pi_3_illustration.svg)

<sup>39</sup>Quelle: 25.03.2021, [https://commons.wikimedia.org/wiki/File:Raspberry\\_Pi\\_3\\_illustration.svg](https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_illustration.svg)

---

## 6 PERFORMANCETESTS

Das Produkt auf Herz und Niere zu testen war die Endstufe für die HSS-Blackbox. Für die Tests wurde ein zweiter Prototyp gedruckt und dieser wurde vollständig aufgesetzt und durchgetestet. Dabei wurden Belastungstests im Bereich der Thermik, und der Mechanik durchgeführt. Es wurden demnach verschiedene Testphasen definiert und diese systematisch dokumentiert,

Der thermische Teil des Performancetests lag darin, Extremverhältnisse bezüglich der Umgebungstemperatur auszutesten, um zu sehen wie sich das System verändert beziehungsweise wie das Material und die Komponenten auf die neue Umgebung reagieren. Dabei wurde die HSS-Blackbox (voll aufgesetzt) in einen Ofen gelegt und die Temperatur wurde langsam hochgedreht. Dabei wurde die Temperatur in 20er Schritten erhöht - startend von 30 Grad Celsius. Jeder Temperaturstand wurde mindestens 10 Minuten angehalten und das HSS wurde aus dem Ofen genommen, um die Auswirkung direkt zu begutachtet. Bei den Temperaturen 30, 50, 70 und 90 Grad Celsius waren keine sichtbaren Einschlüsse, Schmelzungen oder Erweichungen sichtbar. Ab 70 bis 80 Grad Celsius würde PLA schon anfangen zu erweichen. Ab 100 Grad Celsius wurde die Temperatur des Ofens in kleineren Schritten erhöht. Ab circa 105 bis 115 Grad Celsius war eine Erweichung der Oberfläche sichtbar, aber das HSS war immernoch standhaft. Erst bei einer weiteren Erhöhung der Temperatur fing die Oberfläche an sich regelrecht zu verflüssigen. Da diese Temperatur niemals im Alltag erreicht werden kann, sehen wir die thermische Standhaftigkeit des Home-Security-Systems als vollen Erfolg - zu verdanken: ABS und der Gehäusewandstärke. Der zweite Teil des thermischen Performancetests lag darin, das HSS im Schnee zu betreiben - demnach war ein langer Betrieb unter 0 Grad Celsius auch kein Problem für das Gerät und das HSS zeigte keine Performanceverluste. Dabei wurde geachtet, dass keine Feuchtigkeit in das Gerät eindringt, da das HSS nicht wasserabweisend gebaut wurde.

Als weitere Belastung wurden wortwörtlich Gewichte auf das HSS gelegt. Es fing an mit kleinen Gewichten von 10kg. Hierbei überlebte die Blackbox und wir erhöhten auf 25kg. 10 Minuten später wurde die Belastung auf satte 40kg fast verdoppelt. Auch nach dieser Belastung wurden keine Merkmale der Zerstörung wahrgenommen. Nach einer weiteren Weile wurden noch zusätzliche Gewichte draufgelegt, um zu schauen, wann das HSS einbricht. Nachdem wir keine übrigen Gewichte mehr hatten, alle hiermit auf dem HSS waren, kamen wir zum Entschluss,

---

dass die Blackbox unglaublich standhaft und eine extremst hohe Materialfestigkeit aufweist. Nachdem die 65kg (!) abgenommen und die Blackbox begutachtet wurde, waren keine Einschlüsse, keine Kratzer oder keine Dellen sichtbar. Das HSS stand wie zuvor da, als hätten wir keinen Belastungstest mit solch hohen Gewichten vorgenommen!



Abbildung 48: HSS Belastungsstest mit 65kg

---

## 7 ALTERNATIVE LÖSUNGSWEGE

Als Upgradeversuch dieses einfach aufstockbaren Geräts wurden mehrere Ideen diskutiert und auf Sinnmäßigkeit geprüft.

Bessere Komponenten zu benutzen. Das klingt nie falsch. Den DHT11 könnte man mit dem präziseren DHT22 austauschen, der fast ohne Toleranzen arbeitet und sogar von 0-100%, statt 20-80% misst. Es wäre auch möglich, den Bewegungsmelder mit einem weitaus leistungsstärkeren Melder auszutauschen, damit man auch die minimalsten Bewegungen wahrnimmt. Bei der Kamera könnte man auf eine Nachtsichtkamera (statt der Pi Cam Rev 1.3) setzen. Diese Kamera könnte dann auch in der Nacht qualitativ hochwertige Bilder liefern. Diese Kamera-Änderung wäre bei einer 24/7 Überwachung verpflichtend nötig. Die softwaretechnische Implementierung würde sich dabei nahezu garnicht verändern.

Die CPU des Raspberry Pi ist bekanntlich eine Recheneinheit, die Wärme abgibt. Deshalb kann sich der Innenraum der Blackbox nach einer langen Inbetriebnahme sehr stark erhitzen - was sogar unter anderem die gemessenen Temperaturwerte verfälschen könnte. Aus diesem Grund wäre vielleicht eine Lüftung samt Lüftungsschlitzte keine so schlechte Idee dieses Problem zu lösen. Man kann auch, die nicht sichtbare Seite des DHT11 Sensors mit einem hitzeabweisenden Material beziehungsweise mit einem Heat-Sink ausrüsten, um die Verfälschung der gemessenen Werte zu minimalisieren.

Da der Prototyp nun nur über die IP-Adresse zugänglich ist, ist dort auch eine Sicherheitslücke. Jeder der diese Adresse kennt (das heißt wirklich JEDER auf dem Globus) kann auf die Seite zugreifen und somit auch auf die Kamera und dessen Live-Übertragung. Aus diesen Gründen könnte man überlegen eine sichere Login-Page mit einem Benutzernamen und Passwort zu gestalten, damit nur die autorisierten Menschen auf die Seite zugreifen können. Auch eine VPN-Verbindung würde dieses Problem lösen und würde auch eine attraktive Lösung des Problems darstellen.

Nun ein Case: Sagen wir mal Jemand läuft durch das Bild und der Benutzer ist gerade nicht aufmerksam und verpasst diese Auffälligkeit in der Liveübertragung - geschweige denn er ist auf der Subdomain, , wo die Grafen sichtbar sind. Auf der Landing-Page würde er rein gar nichts merken, da man auf der Hauptdomain nichts bezüglich der Bewegungsmeldung oder der Liveübertragung mitbekommt. Dieses Problem kann man so beheben, dass man den Be-

---

wegungsmelder global implementiert und sobald es eine Aktion wahrnimmt, dieser auch auf der Hauptdomain eine Meldung inform einer String oder eine akustische Meldung inform eines lauten Biep-Tons abgibt. Dieses Biep-Ton kann man sowohl am Webserver (Laptop gibt den Ton aus) als auch am Raspberry Pi ausgeben. Zu beachten ist, dass man demnach Lautsprecher (am Raspberry Pi) installieren müsste.

Das Gerät ist nicht wireless. Dabei wird nicht nur die Internetverbindung gemeint, sondern auch die Power-Port (USB-Micro) Verbindung mit dem Laptop oder einer anderen Spannungsquelle. Jetzt ragen zwei Kabel aus dem Gehäuse, die die Mobilität des Geräts stark einschränken. Um dieses Problem zu beheben, kann man das Gerät mit einem WLAN-Adapter und einer Powerbank ausstatten, um das Gerät völlig zu mobilisieren. Das HSS ist nun frei von externen Kabeln und kann überall aufgestellt werden. Diese Integrierung der zwei zusätzlichen Komponenten, würde die Gehäuseplanung neu gestalten, sodass man sich neu überlegen müsste, wie man die Platzierung der zwei neuen Komponenten mit Kombination der alten Komponenten organisiert.

---

## **8 SCHLUSSFOLGERUNG UND PROJEKTERFAHRUNG**

Das Projekt ansich war ein voller Erfolg. Mit diesem einfach upgradebaren Prototyp können Serverräume, Gewächshäuser, Garagen und vieles mehr ganz einfach global überwacht werden, um die Sicherheit dieser Lokationen zu garantieren. Bezüglich der Implementation des Projekts (von Grund auf) kann diese Arbeit als positive Bereicherung der allgemeinen technischen Branchenerfahrung gesehen werden.

Das breite technische Spektrum der HTL Anichstraße und darausfolgend unser Vorwissen plus die eigene Recherche, hat völlig ausgereicht, um so ein Projekt eigenständig und diszipliniert anzugehen. Der rote Faden durch das Projekt wurde am Anfang gelegt und vom Weg zum Ziel wurde nicht abgeschweift. Die Programmierung der Komponenten beziehungsweise die Implementation der Kombination zwischen Komponente und Datenbank sehen wir als erste Grundlage zum erfolgreichen Karrierestart in dieser technischen Branche.

Im Bereich Projektentwicklung beziehungsweise Projektmanagement konnten wir sehr gute Erfolge erzielen. So war es kein Problem ein solches Produkt samt Dokumentation zu zweit herzustellen, da die permanente Absprache zwischen Projektpartner und Projektbetreuer kein Raum für Fehler zugelassen hat. Dieses Wissen in der Projektentwicklung beziehungsweise im Projektmanagement in den letzten fünf Jahren aufzubauen und diese dann im Endspurt an der Diplomarbeit anzuwenden, war für uns eine perfekte Abrundung unserer gesamten HTL-Laufbahn.

Ingesamt sind wir, Toprak Berke und Stojanovic Djordje, mit dem Output dieses Projekts sehr zufrieden. Dennoch werden privat paar Upgrades vorgenommen (siehe Alternative Lösungen Punkt 7). Schließlich lässt sich sagen, wie am Anfang erwähnt: Die Projektplanung, Projektentwicklung und die Zusammenarbeit zwischen Projektpartner und Projektbetreuer ein voller Erfolg.

## 9 PROJEKTTERMINPLANUNG

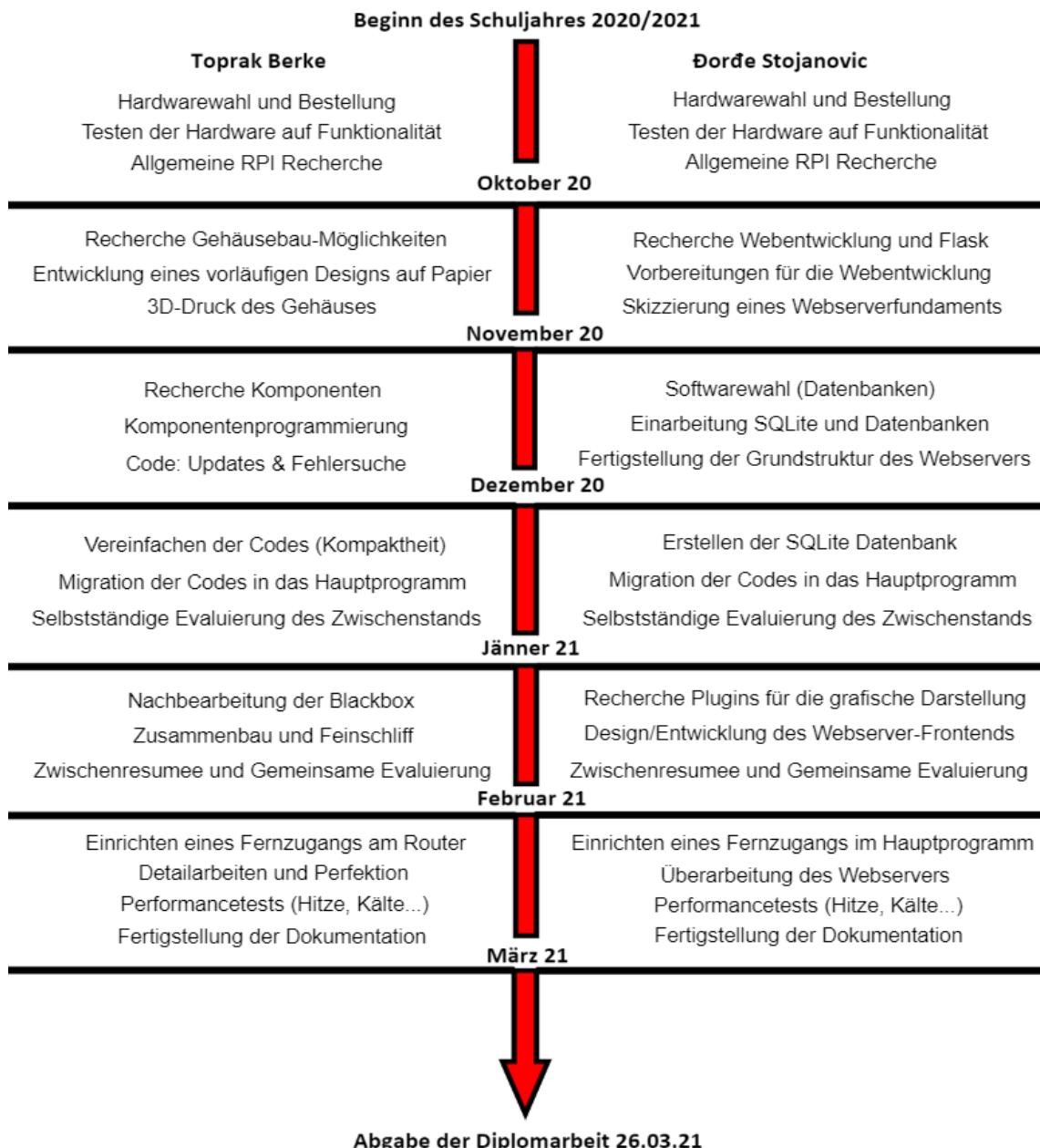


Abbildung 49: Projektdurchführung: "Roter Faden"

---

## 10 MEILENSTEINE

### 10.1 Toprak Berke

Nr.	Meilenstein	Datum
1	Planung der Diplomarbeit, Hardwarewahl und Bestellung	02.10.2020
2	Entwicklung und Produktion des Gehäuses	29.10.2020
3	Aufbau und Konfiguration der Hardware	26.11.2020
4	Datenbeschaffung vom Raspberry System	24.12.2020
5	Fertigstellung der BlackBox	21.01.2021
6	Konfiguration des Netzwerkes für Fernzugang zum Webserver	18.02.2021
7	Verfassung der Diplomschrift	18.03.2021

Tabelle 6: Meilensteine: Toprak Berke

### 10.2 Stojanović Đorđe

Nr.	Meilenstein	Datum
1	Planung der Diplomarbeit, Hardwarewahl und Bestellung	02.10.2020
2	Fundament des Webservers	29.10.2020
3	Fertigstellung der Grundstruktur des Webservers	26.11.2020
4	Implementierung der Daten in den Webserver	24.12.2020
5	Einrichtung graphisches Monitoring & Fernsteuerung des Webservers	21.01.2021
6	Fertigstellung des Webservers	18.02.2021
7	Verfassung der Diplomschrift	18.03.2021

Tabelle 7: Meilensteine: Stojanović Đorđe

---

## 11 ARBEITSNACHWEIS

### 11.1 Toprak Berke (179h)

Datum	Uhrzeit / Dauer [h]	Beschreibung
09.10.2020	18:00 - 22:00 / 4	Brainstorming mit Projektpartner
10.10.2020	14:00 - 19:00 / 5	Allgemeine Recherche
12.10.2020	11:00 - 12:00 / 1	Meeting mit Projektbetreuer zur Überlegung des Projektverlaufs
16.10.2020	18:00 - 21:00 / 3	gemeinsame Hardwarewahl und -bestellung
18.10.2020	11:00 - 16:00 / 5	gemeinsame Softwarewahl
24.10.2020	10:00 - 14:00 / 4	Testen der Hardware auf Funktionalität
25.10.2020	13:00 - 19:00 / 6	Allgemeine Raspberry Pi Recherche
01.11.2020	10:30 - 14:30 / 4	Recherche Gehäusebau-Möglichkeiten
03.11.2020	17:00 - 00:00 / 7	Entwicklung von 22 Blackbox-Skizzen auf Papier
07.11.2020	12:00 - 17:00 / 5	Implementierung der gewählten Skizze in AutoCad
08.11.2020	12:30 - 15:30 / 3	Recherche 3D-Druck (Software, Tricks...)
15.11.2020	20:00 - 00:00 / 4	Übertragung der Zeichnung in eine .stl Datei (Catia)
23.11.2020	18:30 - 19:30 / 1	Meeting mit INNIO Jenbacher GmbH
24.11.2020	08:00 - 12:00 / 4	Produktion des Gehäuses mit einem 3D Drucker
06.12.2020	10.00 - 14:00 / 4	vertiefende Recherche über Komponenten (Aufbau, Funktion, Datasheets)
14.12.2020	10:30 - 11:30 / 1	Meeting mit Projektpartner und -betreuer
15.12.2020	16:00 - 22:00 / 6	Komponentenprogrammierung
19.12.2020	10:00 - 18:00 / 8	Testing, Updating und Fehlersuche der Codes
20.12.2020	13:30 - 16:30 / 3	Vereinfachen der Codes (Verständlichkeit & Kompaktheit)
27.12.2020	11:00 - 13:00 / 4	Dokumentation der bisherigen Arbeit
17.01.2021	12:00 - 17:00 / 5	Meeting mit Projektpartner (Migration der Codes)
23.01.2021	09:00 - 12:00 / 3	Übergabe, Dokumentation & Erklärung des Codes
30.01.2021	10:00 - 15:00 / 5	Selbständige Evaluierung des Zwischenstandes
06.02.2021	13.00 - 16.00 / 3	Überlegungen zur Komponentenbefestigung
11.02.2021	13.00 - 17:00 / 4	Blackbox-Nachbearbeitung in der Werkstatt
12.02.2021	08:30 - 11:30 / 3	Zusammenbau und Feinschliff (Blackbox)
13.02.2021	09:00 - 16:00 / 7	Zwischenresumee und gemeinsame Evaluierung
19.02.2021	19:00 - 21:00 / 2	Meeting mit Projektpartner und -betreuer
01.03.2021	17:00 - 20:00 / 3	Vergewisserung der richtigen HTTPS Settings
06.03.2021	10:30 - 10:40 / 4	Portforwarding - Freischaltung und Konfiguration
07.03.2021	10:00 - 17:00 / 7	DynDNS(VPN)-Versuch - Fehlgeschlagen
10.03.2021	17:00 - 22:00 / 5	Detailarbeiten und Perfektion
11.03.2021	10:30 - 10:40 / 4	Performancetests (Hitze, Kälte, Gewicht)
20.03.2021	xx:xx - xx:xx / 35	Verfassen der Diplomschrift über mehrere Tage
25.03.2021	17:00 - 00:00 / 7	Fertigstellung der Diplomarbeit beziehungsweise der Dokumentation

Tabelle 8: Arbeitsnachweis: Toprak Berke

---

## 11.2 Stojanović Đorđe (176h)

Datum	Uhrzeit / Dauer [h]	Beschreibung
09.10.2020	18:00 - 22:00 / 4	Brainstorming mit Projektpartner
10.10.2020	14:00 - 19:00 / 5	Allgemeine Recherche
12.10.2020	11:00 - 12:00 / 1	Meeting mit Projektbetreuer zur Überlegung des Projektverlaufs
16.10.2020	18:00 - 21:00 / 3	gemeinsame Hardwarewahl und -bestellung
18.10.2020	11:00 - 16:00 / 5	gemeinsame Softwarewahl
24.10.2020	10:00 - 14:00 / 4	Testen der Hardware auf Funktionalität
25.10.2020	13:00 - 19:00 / 6	Allgemeine Raspberry Pi Recherche
04.11.2020	14:00 - 18:00 / 4	Recherche Webentwicklung und Flask
06.11.2020	18:00 - 20:00 / 2	Installation der notwendigen Software
08.11.2020	11:00 - 17:00 / 6	Wiederholung der Python Grundlagen
14.11.2020	13:00 - 20:00 / 7	Vertiefende Wiederholung der HTML, CSS und JS Grundlagen
15.11.2020	10:00 - 15:00 / 5	Studieren der Kernelemente von Flask
28.11.2020	11:00 - 13:00 / 2	Große Überlegungen zur Softwaretechnischen Umsetzung des Webservers
04.12.2020	18:00 - 19:00 / 1	Wahl der erweiterten Software
06.12.2020	10:30 - 15:30 / 5	Studieren von Datenbanken
14.12.2020	10:30 - 11:30 / 1	Meeting mit Projektpartner und -betreuer
19.12.2020	15:00 - 20:00 / 5	Einarbeitung in SQLite
20.12.2020	12:00 - 18:00 / 6	Fertigstellung der Grundstruktur des Webservers
27.12.2020	15:00 - 19:00 / 4	Dokumentation der bisherigen Arbeit
17.01.2021	12:00 - 17:00 / 5	Meeting mit Projektpartner (Migration der Codes)
23.01.2021	11:30 - 13:30 / 2	Einarbeitung in Codes des Projektpartners
24.01.2021	10:00 - 12:00 / 2	Erstellung der SQLite Datenbank
27.01.2021	15:00 - 19:00 / 4	Implementierung des Bewegungssensors
30.01.2021	11:00 - 16:00 / 5	Selbstständige Evaluierung des Zwischenstandes
05.02.2021	17:00 - 22:00 / 5	Fertigstellung des Backends
06.02.2021	10:00 - 14:00 / 4	Recherche bezüglich Frontend
07.02.2021	11:30 - 20:30 / 8	Entwicklung des Frontends
13.02.2021	09:00 - 16:00 / 7	Zwischenresumee und gemeinsame Evaluierung
17.02.2021	16:00 - 19:00 / 3	Überlegungen bezüglich Fernzugriff
19.02.2021	19:00 - 21:00 / 2	Meeting mit Projektpartner und -betreuer
05.03.2021	10:30 - 11:30 / 1	Programminterne Einrichtung des Fernzugriffs
06.03.2021	13:00 - 19:00 / 6	Optimierung und Fertigstellung des Webservers
11.03.2021	12:50 - 16:50 / 4	Performancetests (Hitze, Kälte, Gewicht)
20.03.2021	xx:xx - xx:xx / 35	Verfassen der Diplomschrift über mehrere Tage
25.03.2021	17:00 - 00:00 / 7	Fertigstellung der Diplomarbeit beziehungsweise der Dokumentation

Tabelle 9: Arbeitsnachweis: Stojanović Đorđe

---

## 12 KOSTENTABELLE

ARTIKEL	ANZAHL	EINKAUFSPREIS	BEZUGSORT
RASPBERRY PI 3B+	2	39.99 €	PRIVAT
SD-KARTE	2	3.92 €	<a href="https://www.amazon.de/gp/product/B00KIOWWH2">https://www.amazon.de/gp/product/B00KIOWWH2</a>
DHT11 SENSOR	5	9.06 €	<a href="https://www.amazon.de/gp/product/B07TSF94KD">https://www.amazon.de/gp/product/B07TSF94KD</a>
PI CAM REV 1.3	2	9.99 €	PROJEKTBETREUER
MOTION SENSOR	3	8.05 €	<a href="https://www.amazon.de/gp/product/B00R2U8LLG">https://www.amazon.de/gp/product/B00R2U8LLG</a>
MICRO-USB-KABEL	2	~ 5 €	PRIVAT
JUMPERKABELSET	1	~ 5 €	PRIVAT
LAN-KABEL	2	~ 10 €	PRIVAT
SCHRAUBEN M2	24	~ 5 €	HTL
SCHRAUBEN M3	8		HTL
MUTTERN M2	24		HTL
SPRENGRinge M2	24		HTL
DRUCKKOSTEN PRO BLACKBOX		~ 15 €	
GESAMTKOSTEN PRO BLACKBOX		~ 85.49 €	

Abbildung 50: Kostentabelle

# Abbildungsverzeichnis

1	Webserver-Dashboard . . . . .	v
2	Webserver-Liveübertragung . . . . .	vi
3	Fertiges Produkt . . . . .	vii
4	Raspberry Pi 3B Plus Platine . . . . .	3
5	OmniVision 5647 Blockdiagramm . . . . .	5
6	Raspberry Pi Cam Rev 1.3 Front . . . . .	6
7	Motion Sensor ohne Abdeckung . . . . .	7
8	DHT 11 ohne Abdeckung . . . . .	8
9	Pi Cam Rev 1.3 . . . . .	20
10	HS-SR501 Bewegungsmelder . . . . .	21
11	DHT 11 Sensor mit Outputplatine . . . . .	22
12	Basis: Lizenz . . . . .	25
13	imageresetup.exe ist gestartet . . . . .	27
14	Raspberry OS Imager wird installiert . . . . .	28
15	Raspberry OS Imager ist installiert . . . . .	28
16	OS Imager Dashboard . . . . .	29
17	OS Imager Dashboard: OS Auswahl . . . . .	30
18	OS Imager Dashboard: SD-Karten Auswahl . . . . .	30
19	OS Imager Dashboard: Flashen beendet . . . . .	31
20	Raspberry Pi: Erfolgreiche Installation des Betriebssystems . . . . .	31
21	Raspberry Pi: Betriebssystem-Setup gestartet . . . . .	32
22	Raspberry Pi: Betriebssystem-Setup fertig . . . . .	33
23	PI Cam Installation: Navigierung zum Konfigurationsmenü . . . . .	36
24	PI Cam Installation: Aktivierung der Kamera . . . . .	37
25	SQLite3 Datenbank: Test der Tabelle in der Datenbank . . . . .	40
26	DHT 11 Sensor: Testprogramm Ausgabe . . . . .	42
27	Hauptprogramm: Ordnerstruktur . . . . .	46
28	Hauptprogramm: Finale Ordnerstruktur . . . . .	62
29	Fernzugriff: Portweiterleitung . . . . .	70
30	Fernzugriff: Web-Zertifikat im Detail . . . . .	71
31	Stratatys Fortus 900 MC HighTech-Industriedrucker . . . . .	73
32	Gehäusebau: Skizze 1 . . . . .	75
33	Gehäusebau: Skizze 2 . . . . .	76
34	Gehäusebau: Skizze 3 . . . . .	77

---

35	Raspberry Pi: Dimensionen	78
36	AutoCad-Plot: Gesamtplot	79
37	AutoCad-Plot: Unterseite	80
38	AutoCad-Plot: Deckel	81
39	AutoCad-Plot: Seiten (Links und Rechts)	82
40	AutoCad-Plot: Vorderseite	83
41	AutoCad-Plot: Rückseite	84
42	STL-Design: Deckel	85
43	STL-Design: Basis	86
44	3D-gedrucktes Gehäuse	87
45	Nachbearbeitung: Bohrungen	88
46	Nachbearbeitung: Power-Port	89
47	Gesamtschaltung ohne Pi Cam, 25.03.2021	90
48	HSS Belastungsstest mit 65kg	92
49	Projektdurchführung: "Roter Faden"	96
50	Kostentabelle	100

---

## **Tabellenverzeichnis**

1	Unterschied: Compiler - Interpreter . . . . .	9
2	Basis: Verwendungszwecke und Unterschiede . . . . .	26
3	PIR Motion Sensor: Programmfunction . . . . .	34
4	DHT 11: Programmfunctionen . . . . .	43
5	Hauptprogramm: Programmfunctionen . . . . .	47
6	Meilensteine: Toprak Berke . . . . .	97
7	Meilensteine: Stojanović Đorđe . . . . .	97
8	Arbeitsnachweis: Toprak Berke . . . . .	98
9	Arbeitsnachweis: Stojanović Đorđe . . . . .	99

---

## Codeverzeichnis

1	PIR Sensor: imports und Vorkonfiguration . . . . .	35
2	PIR Sensor: bewegungScannen() . . . . .	35
3	DHT 11: Testprogramm . . . . .	42
4	DHT 11: imports . . . . .	43
5	DHT 11: datenSpeichern(temperatur, luftfeuchtigkeit) . . . . .	44
6	DHT 11: datenAnzeigen() . . . . .	44
7	DHT 11: datenAuslesen() . . . . .	45
8	DHT 11: main() . . . . .	45
9	Hauptprogramm: imports . . . . .	49
10	Hauptprogramm: Initialisierungen . . . . .	50
11	Hauptprogramm: auslesenLetzterDaten() . . . . .	50
12	Hauptprogramm: pruefeGrenzen(temperaturen, luftfeuchtigkeiten) . . . . .	51
13	Hauptprogramm: auslesenHistorischerDaten(anzahlAbtastungen) . . . . .	52
14	Hauptprogramm: anzahlReihen() . . . . .	53
15	Hauptprogramm: abtastungsPeriode() . . . . .	53
16	Hauptprogramm: Globalisierung bedeutender Variablen . . . . .	54
17	Hauptprogramm: index() . . . . .	55
18	Hauptprogramm: my_form_post() . . . . .	56
19	Hauptprogramm: tempearturdiagramm() . . . . .	57
20	Hauptprogramm: luftfeuchtigkeitsdiagramm() . . . . .	58
21	Hauptprogramm: kameraseiteUpdate() . . . . .	59
22	Hauptprogramm: kameraGenerator() . . . . .	59
23	Hauptprogramm: def liveuebertragung() . . . . .	60
24	Hauptprogramm: Erstellen von cert.pem und key.pem . . . . .	61
25	Hauptprogramm: Umsetzung des Fernzugriffs im Hauptprogramm . . . . .	61
26	Hauptprogramm: index.html - header . . . . .	63
27	Hauptprogramm: index.html - body teil 1 . . . . .	64
28	Hauptprogramm: index.html - body teil 2 . . . . .	65
29	Hauptprogramm: index.html - body teil 3 . . . . .	66
30	Hauptprogramm: kamera.html - head . . . . .	67
31	Hauptprogramm: kamera.html - body . . . . .	68
32	Hauptprogramm: style.css . . . . .	69

# Literaturverzeichnis

1	Quelle: Vgl. - 17.03.2021, <a href="https://www.elektronik-kompendium.de/sites/com/1904221.htm">https://www.elektronik-kompendium.de/sites/com/1904221.htm</a>	4
2	Quelle: 17.03.2021, <a href="https://www.imagequalitylabs.com/PDFs/OmniVision_OV5647.pdf">https://www.imagequalitylabs.com/PDFs/OmniVision_OV5647.pdf</a>	5
3	Quelle: Vgl. - 28.02.2021, <a href="https://docs.rs-online.com/2888/0900766b8127db0a.pdf">https://docs.rs-online.com/2888/0900766b8127db0a.pdf</a>	5
4	Quelle: Vgl. - 28.02.2021, <a href="https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html">https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html</a>	7
5	Quelle: Vgl. - 20.03.2021, <a href="https://html.alldatasheet.com/html-pdf/14901/PERKINELMER/LHI778/605/1/LHI778.html">https://html.alldatasheet.com/html-pdf/14901/PERKINELMER/LHI778/605/1/LHI778.html</a>	7
6	Quelle: Vgl. - 21.03.2021, <a href="https://honey-pi.de/dht22-bzw-dht11-verwenden/">https://honey-pi.de/dht22-bzw-dht11-verwenden/</a>	8
7	Quelle: Vgl. - 20.03.2021, <a href="https://www.alldatasheet.com/datasheet-pdf/pdf/1132088/ETC2/DHT11.html">https://www.alldatasheet.com/datasheet-pdf/pdf/1132088/ETC2/DHT11.html</a>	8
8	Quelle: 21.03.2021, <a href="https://www.geeksforgeeks.org/difference-between-compiler-and-interpreter/">https://www.geeksforgeeks.org/difference-between-compiler-and-interpreter/</a>	9
9	Quelle: 22.03.2021, <a href="https://javajee.com/summary-of-object-oriented-programming-oop-concepts-in-java">https://javajee.com/summary-of-object-oriented-programming-oop-concepts-in-java</a>	10
10	Quelle: Vgl. - 22.03.2021, <a href="https://www.python.org/doc/essays/blurb/">https://www.python.org/doc/essays/blurb/</a>	10
11	Quelle: Vgl. - 17.01.2021, <a href="https://wsgi.readthedocs.io/en/latest/what.html">https://wsgi.readthedocs.io/en/latest/what.html</a>	10
12	Quelle: Vgl. - 17.01.2021, <a href="https://palletsprojects.com/p/flask/">https://palletsprojects.com/p/flask/</a>	10
13	Quelle: Vgl. - 17.01.2021, <a href="https://pediaa.com/what-is-the-difference-between-markup-language-and-programming-language/">https://pediaa.com/what-is-the-difference-between-markup-language-and-programming-language/</a>	11
14	Quelle: Vgl. - 17.01.2021, <a href="https://www.javatpoint.com/what-is-html">https://www.javatpoint.com/what-is-html</a>	12
15	Quelle: Vgl. - 17.01.2021, <a href="https://www.javatpoint.com/what-is-css">https://www.javatpoint.com/what-is-css</a>	12
16	Quelle: Vgl. - 17.01.2021, <a href="https://www.w3schools.com/whatis/whatis_js.asp">https://www.w3schools.com/whatis/whatis_js.asp</a>	12
17	Quelle: Vgl. - 17.01.2021, <a href="https://www.oracle.com/database/what-is-database/">https://www.oracle.com/database/what-is-database/</a>	13
18	Quelle: Vgl. - 17.01.2021, <a href="https://www.sqlite.org/index.html">https://www.sqlite.org/index.html</a>	13
19	Quelle: Vgl. - 17.01.2021, <a href="https://bit.ly/3w1OpEI">https://bit.ly/3w1OpEI</a> (Link mit bitly.com gekürzt)	14
20	Quelle: Vgl. - 17.01.2021, <a href="https://www.forcepoint.com/de/cyber-edu/osi-model">https://www.forcepoint.com/de/cyber-edu/osi-model</a>	14
21	Quelle: Vgl. - 19.01.2021, <a href="https://juni.pr/3flSRlu">https://juni.pr/3flSRlu</a> (Link mit bitly.com gekürzt)	16
22	Quelle: Vgl. - 19.01.2021, <a href="https://de.wizcase.com/blog/portweiterleitung-was-ist-es/">https://de.wizcase.com/blog/portweiterleitung-was-ist-es/</a>	16
23	Quelle: Vgl. - 19.01.2021, <a href="https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https">https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https</a>	16
24	Quelle: Vgl. - 22.03.2021, <a href="https://www.autodesk.de/solutions/3d-printing">https://www.autodesk.de/solutions/3d-printing</a>	17
25	Quelle: Vgl. - 22.03.2021, <a href="https://www.prototec.de/3d-druck-materialien">https://www.prototec.de/3d-druck-materialien</a>	17
26	Quelle: Vgl. - 22.03.2021, <a href="https://www.prototec.de/wp-content/uploads/2021/01/PROTOTEC-3D-Druck-Datenblatt-ABS-FDM.pdf">https://www.prototec.de/wp-content/uploads/2021/01/PROTOTEC-3D-Druck-Datenblatt-ABS-FDM.pdf</a>	18
27	Quelle: 22.03.2021, <a href="https://www.stratasys.com/de/materials/search/abs-m30">https://www.stratasys.com/de/materials/search/abs-m30</a>	18
28	Quelle: Vgl. - 28.02.2021, <a href="https://docs.rs-online.com/2888/0900766b8127db0a.pdf">https://docs.rs-online.com/2888/0900766b8127db0a.pdf</a>	20
29	Quelle: Vgl. - 28.02.2021, <a href="https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html">https://pdf1.alldatasheet.com/datasheet-pdf/view/1131987/ETC2/HC-SR501.html</a>	21
30	Quelle: Vgl. - 28.02.2021, <a href="https://www.alldatasheet.com/datasheet-pdf/pdf/1132088/ETC2/DHT11.html">https://www.alldatasheet.com/datasheet-pdf/pdf/1132088/ETC2/DHT11.html</a>	22
31	Quelle: 28.02.2021, <a href="https://www.guru99.com/flask-vs-django.html">https://www.guru99.com/flask-vs-django.html</a>	23
32	Quelle: 22.03.2021, <a href="https://github.com/Mjrovai/Video-Streaming-with-Flask">https://github.com/Mjrovai/Video-Streaming-with-Flask</a>	25
33	Quelle: 10.03.2021, <a href="https://github.com/adafruit/Adafruit_Python_DHT">https://github.com/adafruit/Adafruit_Python_DHT</a>	41
34	Quelle: 21.03.2021, <a href="https://bit.ly/3w1pu47">https://bit.ly/3w1pu47</a> (Link mit bitly.com gekürzt)	59

---

35	Quelle: 17.03.2021, <a href="https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https">https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https</a> . . . . .	61
36	Quelle: 17.03.2021, <a href="https://toorshia.github.io/justgage/">https://toorshia.github.io/justgage/</a> . . . . .	62
37	Quelle: 13.03.21, <a href="https://bit.ly/3roYmZk">https://bit.ly/3roYmZk</a> (Link mit bitly.com gekürzt) . . . . .	78
38	Quelle: 25.03.2021, <a href="https://upload.wikimedia.org/wikipedia/commons/a/a3/Raspberry_Pi_3_illustration.svg">https://upload.wikimedia.org/wikipedia/commons/a/a3/Raspberry_Pi_3_illustration.svg</a> . . . . .	90
39	Quelle: 25.03.2021, <a href="https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_illustration.svg">https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_illustration.svg</a> . . . . .	90