Abschlussprüfung Sommer 2024

IT-Systemelektroniker

Dokumentation zur betrieblichen Projektarbeit

SCHADENSERFASSUNGSSYSTEM

Ausstattung des Warenein- und Ausgangs mit Kamerasystemen zur Erfassung von Schäden bei Kundengeräten

Abgabedatum: Abgabeort

Prüfungsbewerber

Paul Hohmann

Ilextwiete 1

22111 Hamburg

Ausbildungsbetrieb

Enfinitec Germany

GmbH

Kornkamp 4

22926 Ahrensburg

Inhaltsverzeichnis

1. Einleitung

Die folgende Projektdokumentation befasst sich mit der Umsetzung eines Kamerasystems im Warenein- und Ausgang des Repair-centers. Das Projekt wurde vom mimimim bei der Enfinitec Germany GmbH in Ahrensburg durchgeführt. Projektverantwortlicher für die Planung und Durchführung war Paul Hohmann, Auszubildender zum IT-Systemelektroniker. Ansprechpartner zu dem Projekt war Torben Slawinski, Ausbilder der IT-Systemelektroniker. Alle Begriffe, die in der Dokumentation **fett** und **grün** geschrieben sind, werden anschließend im Glossar genauer erläutert.

1.1 Projektumfeld

Die Enfinitec Germany GmbH, ist eine Tochtergesellschaft der Acer Computer GmbH und hat ihren Sitz in Ahrensburg. An diesem Standort sind etwa 210 Mitarbeiter beschäftigt. Es ist ein Unternehmen, das sich auf After-Sales-Services für in- und ausländische Anbieter von elektronischen, mechanischen oder anderen Geräten spezialisiert hat. Sie bietet eine breite Palette von Dienstleistungen an, die von Kundendienstleistungen bis hin zu High-End-Lösungen und schneller bequemer Servicebereitstellung reichen. Die eingehenden Kundengeräte kann man in 5 Kategorien unterteilen:

- Notebooks und Tablets
- Monitore und Beamer
- Desktop PC’s
- Staubsauger und Haushaltsgeräte
- **E-Mobility**

1.2 Projektziel

Das Hauptziel dieses Projekts ist die Implementierung einer effizienten und effektiven Lösung zur Dokumentation und Nachverfolgung von Stoßschäden, Kratzern und Dellen an Kundengeräten. Dies wird erreicht durch die Nutzung einer Fotobox, die in der Lage ist, hochauflösende Bilder der Geräte zu erfassen, sowohl bei der Ein- als auch bei der Ausgangskontrolle. Diese Fotobox besteht aus einem **Raspberry Pi** und einer daran angeschlossenen Webcam. Diese Bilder werden unter der im Betrieb genutzten „Case-ID“ gespeichert. Einer 9-stelligen Identifikationsnummer die mit der Seriennummer zusammenhängt. Die Bilder werden automatisch von dem Raspberry Pi an einen Datei-Server geschickt, so dass berechtigte Mitarbeiter auf die Fotos Zugriff haben.

1.3 Projektbegründung

Da es häufiger zu Transportschäden bei Kundengeräten durch den Versand kommt, sollen diese Schäden genauer protokolliert werden und dieses möglichst nicht den normalen Betrieb behindern.
Durch die genaue Aufnahme der Kundengeräte durch die Kamera werden potenzielle Schäden, die während des Transports oder der Reparatur auftreten können, genau identifiziert und dokumentiert. Dies verbessert nicht nur den Kundenservice durch erhöhte Transparenz, sondern ermöglicht es auch, die internen Prozesse kontinuierlich zu überprüfen und zu verbessern, um zukünftige Schäden zu minimieren.
Insgesamt wird durch dieses Projekt eine nachhaltige Verbesserung der Servicequalität und Kundenzufriedenheit erreicht.

1.4 Projektschnittstellen

Die Fotobox besteht aus einem Raspberry Pi Einplatinencomputer, einer daran angeschlossenen Webcam und einem Barcode-Scanner. Um eine stabile Verbindung zum, im **Intranet** von Enfinitec verbundenen Datei-Server, herzustellen, wird der Raspberry Pi über ein **Ethernet** Kabel verbunden. Zum Interagieren mit dem Raspberry Pi gibt es einen Taster und zwei Leuchtdioden (LED's).

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projekts wurden von der IHK 35 Zeitstunden vorgegeben, die vor Projektbeginn auf verschiedene Phasen verteilt wurden.

Projektphase	Geplante Zeit
Ist-Analyse	4 h
Sollkonzept	5 h
Angebot, Vergleich	4 h
Bildaufnahme-system bauen	7 h
PC einrichten, Serveranbindung einstellen	7 h
Dokumentation	8 h
Gesamt	**35 h**

2.2 Ressourcenplanung

Für die Umsetzung dieses Projektes werden folgende Ressourcen benötigt. Diese sind unterteilt in: Hardware Fotobox, Hardware Projektentwicklung und Software.

2.2.1 Hardware Fotobox

Diese Tabelle beinhaltet die Ressourcen, die verwendet wurden, um zwei vollständige Fotoboxen zu bauen. Die Kosten die mit einem *Sternchen versehen sind nicht anfallende Kosten, da diese Ressourcen bereits vorhanden waren und aus dem Bestand der Enfinitec Germany GmbH zur Verfügung gestellt wurden.

Ressourcen	Kosten	Quantität
Raspberry Pi 4 (2GB)	47,80€	2
Acer FHD Webcam	*19,95€	2
Inateck USB Barcode Scanner	*29,99€	2
Micro-SD Karte, 32GB	5,48€	2
USB-C Netzteil	7,99€	2
Drucktaster	0,11€	2
Jumperkabel, Stecker/Buchse	0,15€	12
Leuchtdioden, rot, 3mm	0,14€	4
Widerstand 330 Ohm	0,07€	4
Kartonbox 10cmx15cmx5cm	*0,00€	2
Gesamtkosten:	**125,40€**	

2.2.2 Hardware Projektentwicklung

Hier werden alle Ressourcen aufgelistet, die benötigt wurden, um dieses Projekt zu entwickeln und umzusetzen.

Ressourcen	Beschreibung/Verwendung
Acer Travelmate P614	Arbeitslaptop
SD-Karten Lesegerät	Beschreiben der SD-Karte für die Raspberry Pi‘s
Lötstation	Löten der Bauteile

2.2.3 Software

Ressourcen	Beschreibung/Verwendung
Windows 10	Betriebssystem
Raspberry Pi Imager	Schreiben des Raspberry Pi Betriebssystems auf die SD-Karten
Raspberry Pi OS Lite 64 Bit	Raspberry Pi Betriebssystem
Visual Studio Code	Programmieren des Python Scripts
Microsoft Word	Schreiben der Dokumentation

3 Analysephase

Nach Abschluss der Planungsphase beginnt die Analysephase. In dieser Phase wird der Zustand, bevor das Projekt umgesetzt wurde, analysiert und der jetzige Zustand nach dem Projekt. Darüber hinaus wird der allgemeine wirtschaftliche Aspekt des Projekts untersucht und dargestellt.

3.1 Ist-Analyse

Die Ist-Analyse beschreibt den Zustand der Arbeitsplätze und Prozesse vor dem Durchsetzen des Projektes. Die Analyse ist in drei Phasen unterteilt. Die Phasen fangen chronologisch mit dem Eintreffen der Kundengeräte an und verfolgen diese durch den Wareneingang, die Reparaturphase, und schließlich den Warenausgang und Versand.

3.1.1 Wareneingang

Der Prozess startet mit dem Eintreffen der defekten Geräte im Repair-Center. Wir konzentrieren uns hier auf den Teil des Wareneingangs der hauptsächlich Notebooks, Monitore sowie Desktop-PCs für die Reparatur vorbereitet. Hierzu werden die Verpackungen, in denen die Kundengeräte eingeschickt werden, geöffnet und in eine neutrale Verpackung umverpackt, um die Reparatur für die Techniker zu optimieren und einen schnelleren Reparaturablauf zu gewährleisten. Dazu wird dann der ausgedruckte Reparaturzettel gelegt, der die Fehlerbeschreibung, Name des Kunden beziehungsweise Händlers, Seriennummer des Geräts und eine 9-stellige Identifikationsnummer beinhaltet. Diese wird hier als Case-ID betitelt. Diese Case-ID wird durch Label mit einem Barcode an den Karton des Kundengerätes geklebt, und kann durch das hauseigene Programm **Contact Center Suite** eingescannt werden. Hier sind alle bekannten Daten zu dem Gerät hinterlegt.

3.1.2 Reparaturphase

Alle vorbestehenden Schäden am Gerät werden hier schriftlich protokolliert und in der Contact Center Suite abgespeichert. Bei großen Schäden wie einem gebrochenen Display oder gebrochenen Gehäuseteilen machen die Techniker Fotos. Anhand dieser kann ein Kostenvoranschlag erstellt werden und bei Fragen von Kunden ist alles mit Fotos dokumentiert. Das Kundengerät wird dann repariert.

3.1.3 Warenausgang und Versand

Nach der Reparatur werden die Geräte für den Versand verpackt. Sollten hier weitere Schäden festgestellt werden, die nicht von den Technikern bei der Reparaturphase protokolliert wurden, werden diese nachträglich fotografiert und in einem Ordner auf einem, von den Mitarbeitern zugänglichen, Dateiserver abgelegt.
Sollten Kunden, nachdem sie ihr repariertes Gerät empfangen haben, einen Schaden feststellen, ist im optimalen Fall alles dokumentiert, so dass besser festgestellt werden kann, ob der Schaden bei der Reparatur oder beim Transport erfolgt ist.

3.2 Soll-Analyse

Die Soll-Analyse beschreibt den aktuellen Zustand, nachdem das Projekt durchgesetzt wurde. Auch diese Analyse wird in mehrere Phasen unterteilt. In der Zieldefinition wird das Projektziel aus der Einleitung noch einmal aufgefasst und detaillierter ausgeführt. Die technischen Anforderungen befassen sich mit der technischen Ausführung des Projekts und beschreibt, wie das System funktioniert.

3.2.1 Zieldefinition

Das Ziel des Projektes besteht darin, es den Mitarbeitern des Waren-ein und -ausgangs zu ermöglichen, die Kundengeräte die sie bearbeiten, einfacher mit Bildern zu protokollieren. Dazu wird am Arbeitsplatz eine Box mit einem Taster und zwei Leuchtdioden platziert. In dieser Box befindet sich der Raspberry Pi, an den ein Barcode-Scanner, ein Ethernet Kabel und die Webcam angeschlossen sind. Die Webcam ist über dem Arbeitsplatz montiert und auf die Arbeitsfläche ausgerichtet. Wenn der Taster betätigt wird, leuchtet eine LED auf, was bedeutet, dass das System startbereit ist. Nun kann mit dem Barcode Scanner die Case-ID eingescannt werden, diese bestimmt den Ordnernamen, in dem die Bilder abgespeichert werden. Nachdem diese eingescannt wurde, leuchtet eine zweite LED auf. Die Kamera wartet nun auf eine weitere Eingabe des Tasters und löst dann aus. Dies wird visuell bestätigt mit dem Aufblinken beider LED-Lampen. Das Bild wird nun in den Dateiserver hochgeladen.

3.2.2 Technische Anforderungen

Der Taster und die Leuchtdioden sind über die **GPIO-Pins** des Raspberry Pi verbunden und werden über diesen auch mit Strom versorgt. Sie sind sichtbar an der Box platziert, werden über ein **Python**-Skript angesteuert und sind die Haupt-ein und -ausgabe Punkte des Systems.

3.3 Wirtschaftlichkeitsanalyse

3.3.1 Projektkosten

Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten und den Kosten der für das Projekt notwendigen Hardwarekomponenten zusammen.
Laut Ausbildungsvertrag verdient ein Auszubildender im dritten Lehrjahr pro Monat *1.300 €* (brutto). Bei *40* Stunden in der Woche ergibt das unter der Formel, die von der Rentenversicherung des Bundes akzeptiert ist, *173,33* Stunden im Monat. Dadurch ergibt sich ein Stundensatz von 7,50€.

$$\frac{40\,\frac{h}{Woche} \times 13\,\frac{Wochen}{3\ Monate}}{3} = 173{,}33\,\frac{h}{Monat}$$

$$\frac{1300\,\frac{€}{Monat}}{173{,}33\,\frac{h}{Monat}} \approx 7{,}50\,\frac{€}{h}$$

Die Durchführungszeit des Projekts beträgt 35 Stunden. Für die Nutzung von Ressourcen wird ein pauschaler Stundensatz von *5 EUR* angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundensatz von *20 EUR* angenommen.

Wie der Tabelle, der Hardwareressourcen für die Fotobox in **2.2.1,** zu entnehmen ist, sind für das Projekt gesamte Kosten von 125,40€ angefallen. In folgender Tabelle sind alle Kosten aufgestellt. Die Zeit wird mit den Kosten pro Stunde, mal benötigten Mitarbeitern multipliziert. Somit kommen wir auf ein Gesamtergebnis von **637,90€.**

Vorgang	Zeit	Kosten / Stunde	Kosten
Entwicklung	35 h	*7,50 € + 5 € = 12,50 €*	*437,50 €*
Abnahme	1 h	*20 € + 5 € = 25,00 €* *x 1 Mitarbeiter*	*25,00 €*
Schulung	1 h	*20 € + 5 € = 25,00 €* *x 2 Mitarbeiter*	*50,00 €*
Hardware	-/	*-/*	*125,40 €*
		Gesamt	***637,90 €***

3.3.2 Amortisationsdauer

Das Projekt ist insbesondere zeitsparend für die Mitarbeiter im Waren-ein und Ausgang sowie die Techniker in der Reparatur, da diese nicht jedes Mal zum Protokollieren von großen Schäden die Kamera rausholen, den Schaden abfotografieren und die Bilder in einen Ordner hochladen müssen. Dies erspart pro Vorgang in etwa 3 Minuten. Ein Techniker macht am Tag im Durchschnitt 12 Kundengeräte, davon haben circa 3 Geräte einen größeren Schaden, welche mit einem Bild protokolliert werden muss.
Somit haben wir eine Zeiteinsparung von 9 Minuten pro Techniker pro Tag.
Bei einer Zeiteinsparung von 9 Minuten pro Techniker pro Tag, ergibt sich folgende Rechnung bei 30 beschäftigten Technikern + 2 beschäftigte Mitarbeiter im Waren-ein und Ausgang:

$$0{,}15\ \frac{h}{\frac{Mitarbeiter}{Tag}} \times\ 32\ Mitarbeiter\ = 4{,}8 \frac{h}{Tag}$$

$$4{,}8 \frac{h}{Tag} \times 220\ Tage = 1056 \frac{h}{Jahr}$$

Dadurch ergibt sich eine jährliche Einsparung von:

$$1056\ h \times (20€ + 5€) = 26.400€$$

Die Amortisationszeit beträgt also:

$$\frac{637{,}90\ €}{26.400 \frac{€}{Jahr}} = 0{,}024\ Jahre = 8{,}76\ Tage \approx \mathbf{9\ Tage}$$

3.3.3 Nutzwertanalyse

In der Nutzwertanalyse werden folgende Modelle des Raspberry Pi's miteinander verglichen:
-**Raspberry Pi 4 2GB, Raspberry Pi 3 B+, Raspberry Pi 3 A+**

Es wird folgender Bewertungsmaßstab verwendet:

1	Sehr schlecht
2	Schlecht
3	Durchschnittlich
4	Gut
5	Sehr gut

In Rot stehen die vergebenen Punkte
In Blau stehen die mit der Gewichtung berechneten Gewichtungspunkte

Bewertungs-kriterien	**Gewichtung**		**Punkte**		**Punkte**		**Punkte**
Modell		Raspberry Pi 4 2GB		Raspberry Pi 3 B+		Raspberry Pi 3 A+	
Preis	40%	47,80€	2 x0,4=0,8	38,50€	3 x0,4=1,2	26,90€	4 x0,4=1,6
Arbeitsspeicher	30%	2 GB	5 x0,3=1,5	1 GB	3 x0,3=0,9	512 MB	2 x0,3=0,6
Schnittstellen	20%	2xUSB 2.0 2xUSB 3.0 Ethernet	5 x0,2=1	4xUSB 2.0 Ethernet	4 x0,2=0,8	1xUSB 2.0	3 x0,2=0,6
Stromverbrauch	10%	5V 3A	2 x0,1=0,2	5V 2,5A	3 x0,1=0,3	5V 2,5A	3 x0,1=0,3
Gesamt	**100%**		**3,5**		**3,2**		**3,1**

Der Raspberry Pi 4 2GB gewinnt hier trotz seines höheren Preises. Durch die 2GB **Arbeitsspeicher** und die hervorragenden Schnittstellen.

4 Durchführung

Bei der Durchführung werden die einzelnen Schritte, die es benötigt **eine** Fotobox zu konfigurieren und aufzubauen erklärt. Da für den jeweils Wareneingang sowie den Warenausgang eine Fotobox gebaut wird.

4.1 Einrichtung des Raspberry Pi's

Für den Raspberry Pi wird eine Micro-SD Karte benötigt, die als Speichermöglichkeit genutzt werden kann und auf welcher das Betriebssystem gespeichert ist. Da der Raspberry Pi nur die Bilder weiterleitet und nicht speichert, genügt hier eine einfache 32 GB SD-Karte. Um das Betriebssystem auf die SD-Karte zu kopieren, wird der **Raspberry Pi Imager** benutzt. Dort wählen wir das **Raspberry Pi OS Lite (64-Bit)** aus. Im Imager können wir direkt einige Einstellungen wie den Usernamen und das Passwort für den Raspberry Pi festlegen. Außerdem kann hier direkt **SSH** eingerichtet werden.

Das konfigurierte Betriebssystem wird auf die SD-Karte kopiert, in den Raspberry Pi gesteckt und dieser durch das Anschließen des USB-C Stromkabels eingeschaltet. Um eine stabile Verbindung zum internen Netzwerk zu gewährleisten, verwenden wir außerdem ein Ethernet Kabel.

4.2 SSH-Verbindung zum Raspberry Pi

Um Ressourcen zu sparen, wird der Raspberry Pi **Headless** betrieben. Um den Raspberry Pi über SSH zu erreichen, brauchen wir als erstes die **IP-Adresse**. Diese finden wir heraus, indem wir den vorher festgelegten Hostnamen **pingen**.

```
paulhohmann@Pauls-Air ~ % ping pi
PING pi.fritz.box (192.168.178.51): 56 data bytes
64 bytes from 192.168.178.51: icmp_seq=0 ttl=64 time=112.965 ms
64 bytes from 192.168.178.51: icmp_seq=1 ttl=64 time=15.227 ms
64 bytes from 192.168.178.51: icmp_seq=2 ttl=64 time=17.751 ms
```

Dieser gibt uns als IP-Adresse des PI's die **192.168.178.51**. Mit dieser Adresse stellen wir jetzt unsere SSH-Anfrage. Dafür benötigen wir den Usernamen und die IP-Adresse. Danach werden wir aufgefordert unser vorher im Raspberry Pi Imager festgelegtes Passwort einzugeben.

```
paulhohmann@Pauls-Air ~ % ssh admin@192.168.178.51
admin@192.168.178.51's password:
Linux pi 6.6.28+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.28-1+rpt1 (2024-04-22) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon May 13 11:14:59 2024 from 192.168.178.20
admin@pi:~ $
```

Nun befinden wir uns im Home-Verzeichnis des Raspberry Pi's. Als erstes können wir mit **sudo apt-get update & sudo apt-get upgrade** den Pi updaten, um Kompatibilität und die neuesten Sicherheitsupdates zur Verfügung zu stellen.

4.3 Anschließen der Kamera

Wir schließen die Webcam über den USB 2.0 Port des Raspberry PI's an und überprüfen die Verbindung mit dem **lsusb** Befehl:

```
admin@pi:~ $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 017: ID 1bcf:28c4 Acer FHD Camera Microphone
Bus 001 Device 016: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Die Kamera wird hier nachvollziehbar unter dem Namen **Acer FHD Camera** angezeigt.

4.4 Installieren der benötigten Software

Für die Webcam benötigen wir die Software **fswebcam** und für das Steuern der USB-Ports **Uhubctl**. Unter dem Punkt „4.7 Uhubctl“ wird dies detaillierter erklärt. Mit folgenden **Sudo** Befehlen werden beide Programme installiert:

```
admin@pi:~ $ sudo apt-get install fswebcam
admin@pi:~ $ sudo apt-get install uhubctl
```

4.5 Erstellen eines Ordners

Um eine Abgrenzung zu schaffen, erstellen wir mit **Mkdir** einen neuen Ordner Camera:

```
admin@pi:~ $ mkdir camera
```

4.6 Fswebcam

Um mit fswebcam ein Bild zu machen, benutzen wir folgenden Befehl:

Fswebcam -r 1920x1080 -p YUYV -S 20 -D 2 -F 2 /home/admin/camera/$FILE_NAME

- Fswebcam befiehlt dem Pi, das diese Software benutzt werden soll.
- r 1920x1080 setzt ein Bildqualität von 1920x1080 Pixeln fest.
- p YUYV setzt den Farbmodus **YUYV** fest.
- -S 20 -D 2 -F 2 setzt weitere Spezifikation fest, um ein optimales Bild zu gewährleisten.
- S 20 überspringt die ersten 20 Bilder der Aufnahme.
- D 2 wartet 2 Sekunden bevor die Bildaufnahme startet, um die Kamera initialisieren zu können.
- F 2 benutzt 2 Bilder und setzt diese zusammen, um Rauschen zu verringern
- /home/admin/camera bestimmt den Pfad, in dem das Bild gespeichert werden soll.
- /$FILE_NAME bestimmt den dynamischen Dateinamen, der im Shellscript festgelegt wird.

4.7 Uhubctl

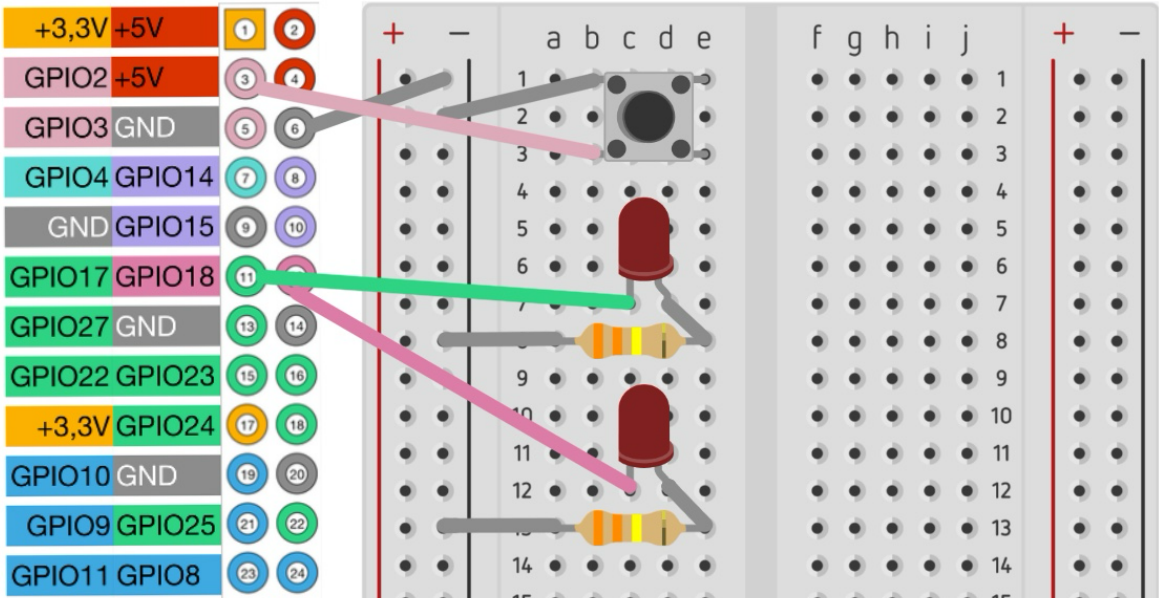
Mit Uhubctl können USB-Ports des Raspberry Pi aus und wieder angeschaltet werden. Dies wird benötigt da die Webcam nach dem ersten Bild einen Fehler ausgibt und kein weiteres Bild mehr gemacht werden kann. Bei der Suche in Foren nach Lösungen, stellt sich heraus das mehrere Leute dieses Problem haben und es wohl ein Problem des Raspberry Pi’s ist. Nach dem physikalischen aus und wieder einstecken der Webcam funktioniert diese wieder, aber nur einmal. Also muss die Webcam wieder ein- und ausgesteckt werden. Durch Uhubctl passiert dies automatisch, indem der Stromfluss zum USB-Port auf dem Raspberry Pi eingestellt wird.

4.8 Dateiserver einrichten

Da bei der Enfinitec Germany GmbH ein bereits vorhandener Dateiserver besteht, habe ich mich an diesem orientiert und erstelle einen neuen Ordner, in dem die Bilder gespeichert werden. In diesem Dateiserver sind außerdem alle Rechte und Zugriffe bereits konfiguriert, so dass jeder Mitarbeiter mit den richtigen Zugriffsrechten auf die hochgeladenen Bilder zugreifen kann.
Ich logge mich auch hier wieder mit SSH in den Server ein. In den Unterordnern des Servers erstelle ich mit Mkdir den Ordner **Pictures**. Um die Bilder vom Raspberry Pi zum Dateiserver zu kopieren, wird der Befehl **scp** verwendet. Dazu wird das Ziel vorher als Variable festgelegt. In diesem Fall stellt der Pi eine Verbindung via SSH her und kopiert das gemachte Bild dorthin.

4.9 Taster und LED‘s an den Raspberry Pi anschließen

Der Taster und die LED’s sind wie folgt mit den GPIO Pins des Raspberry Pi’s verbunden:



Der Taster ist einerseits an den **GPIO-Pin 2** angeschlossen und andererseits mit **GND(Ground)** verbunden. Da jedes Bauteil mit GND verbunden wird, habe ich die Masseleiterbahn des **Breadboards** benutzt. So wird nur ein GND-Pin des Raspberry Pi genutzt. Die erste LED wird an **GPIO-Pin 17** angeschlossen und dann mit einem 330 Ohm Vorwiderstand verbunden, der an GND angeschlossen ist. Die zweite LED wird ebenso mit einem 330 Ohm Widerstand an GND und an **GPIO-Pin 18** angeschlossen. So kann jedes Bauteil einzeln über die GPIO-Pins angesteuert werden. Diese werden dann im Python-Skript definiert.

4.10 Python-Skript

4.10.1 Installieren von Python

Für unser Skript benutzen wir Python da es für Raspberry Pi am besten optimiert ist und wir somit die GPIO-Pins ansteuern können. Dafür wird Python mit folgenden Befehlen von der offiziellen Python-Website heruntergeladen und installiert:

Um Python Version 3.10.0 von der offiziellen Website herunterzuladen, benutzen wir wget:

```
wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tar.xz
```

Danach müssen wir die komprimierte Datei entpacken:

```
tar -xvf Python-3.10.0.tar.xz
```

Nun navigieren wir uns in den extrahierten Ordner und bereiten den Aufbauprozess von Python vor:

```
cd Python-3.10.0
./configure --enable-optimizations
```

Dieser Befehl installiert nun Python:

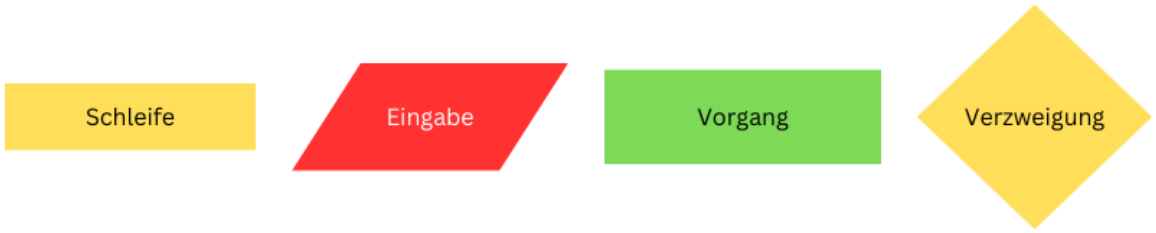
```
make -j 4
sudo make altinstall
```

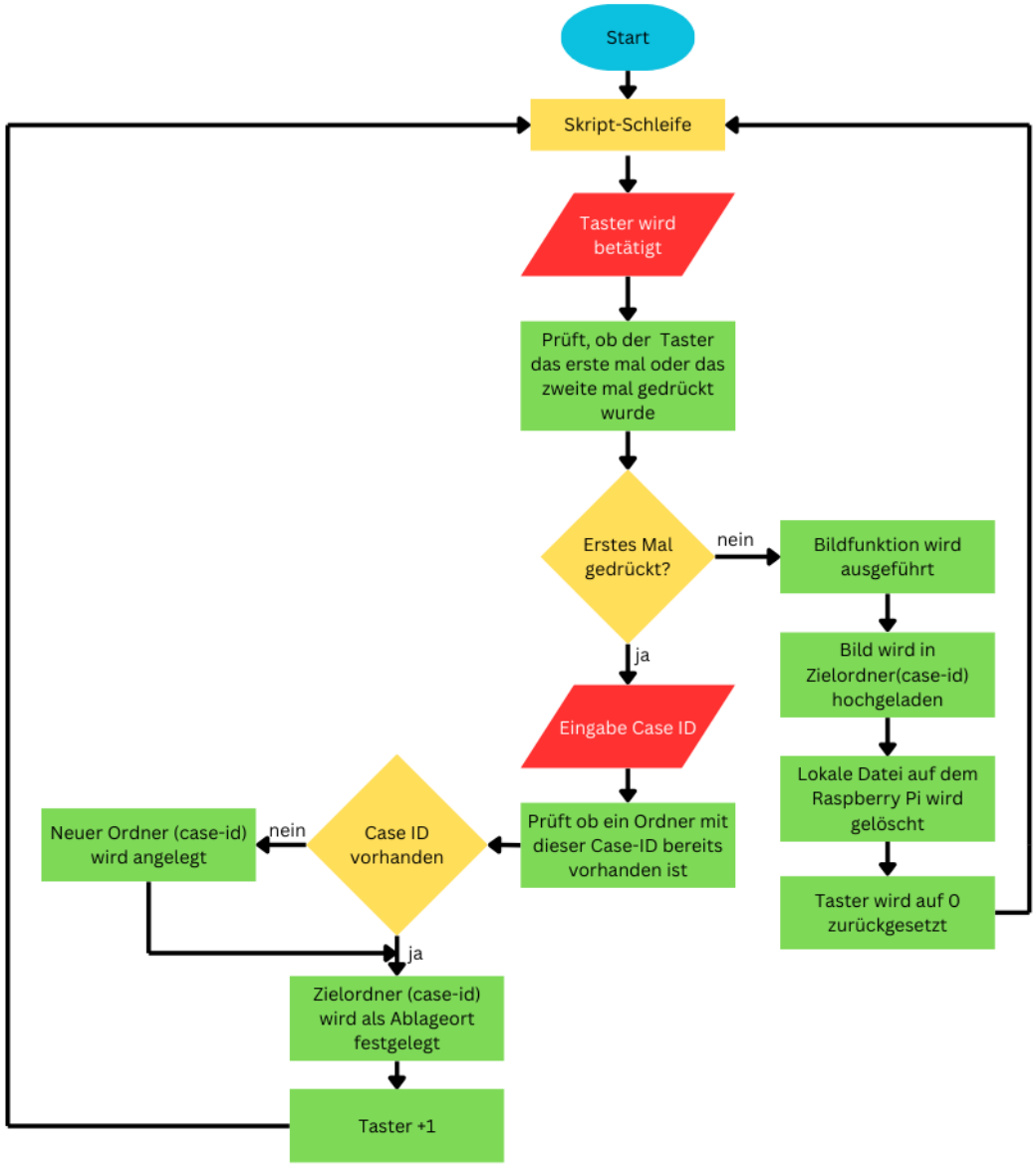
4.10.2 Python-Skript schreiben

Um das Skript zu schreiben, benutzen wir **Visual Studio Code** als Umgebung. Hier verbinden wir uns über ein SSH **Add-on** mit dem Raspberry Pi und können nun unser Python-Skript schreiben. Die Datei benennen wir camera.py.

4.10.3 Programmablaufplan

Der Programmablaufplan beschreibt das Skript und seine Funktionen in einer nachvollziehbaren Struktur. Zur genauen Veranschaulichkeit ist dieser Ablauf stark vereinfacht dargestellt. Man beachte folgende Legende:





4.10.4 Bibliotheken importieren

Als erstes werden mit import die verschiedenen Bibliotheken eingebunden, die für das Python-Skript benötigt werden:
os – Biete Funktionen zur Interaktion mit dem Betriebssystem, wie zum Beispiel Zugriff auf Dateien und Verzeichnisse.
subprocess – enthält Funktionen mit denen Linux Befehle in Python ausgeführt werden können
datetime – Beinhaltet Datums- und Zeitobjekte.
shutil – Stellt Funktionen zum Kopieren, Verschieben und Löschen von Dateien und Verzeichnissen bereit.
gpiozero - Eine Bibliothek zur Steuerung von GPIO-Pins auf dem Raspberry Pi. Es ermöglicht die einfache Steuerung von LEDs, Buttons und anderen Geräten.
signal – Bietet Funktionen zur Behandlung von Signalen, wie das Anhalten von Programmen bei einem bestimmten Signal.
time – Stellt Funktionen zur Zeitmessung und –steuerung bereit.

```
import os
import subprocess
import datetime
import shutil
from gpiozero import Button, LED
from signal import pause
import time
```

4.10.5 Taster und Leuchtdioden definieren

Über die **gpiozero** Bibliothek können wir hier die einzelnen Bauteile benennen und diesen ein GPIO-Pin zuweisen. Danach werden den zugewiesenen Pins eigene Variablen zugeordnet.

```
button_pin = 2
led_one_pin = 17
led_two_pin = 18
led_one = LED(led_one_pin)
led_two = LED(led_two_pin)
button = Button(button_pin)
```

4.10.6 Blinken der Leuchtdioden

Zum Bestätigen von Eingaben sollen die LED's blinken. Diese Funktion akzeptiert zwei Parameter: **led** der die LED darstellt, die blinken soll und **times** was festlegt wie oft die **for-schleife** durchlaufen wird. Nach dem Einschalten der LED wird eine 0,2 Sekunden Verzögerung festgelegt, bis die LED wieder ausgeschaltet wird. Für das Blinken von beiden LED's gleichzeitig, wird das selbe Prinzip angewandt.

```
def flash_led(led, times):
    for _ in range(times):
        led.on()
        time.sleep(0.2)
        led.off()
        time.sleep(0.2)

def flash_both_leds(times):
    for _ in range(times):
        led_one.on()
        led_two.on()
        time.sleep(0.2)
        led_one.off()
        led_two.off()
        time.sleep(0.2)
```

4.10.7 Taster betätigen

Zuerst wird die globale Variable “button_press_count“ festgelegt. Diese verfolgt den Status des Knopfdrucks. Wird dieser wie in der Definition “button_pressed“ gedrückt erhöht sich die Variable um 1. Nun wird mit einer **If-Abfrage** überprüft, ob der “button_press_count“ gleich 1 ist. Wenn ja wird die erste LED eingeschaltet und die Funktion “execute_part1“ ausgeführt. Diese beinhaltet das Skript, in dem die USB-Ports neugestartet werden und die Case-ID eingescannt wird. Danach blinkt die erste LED fünfmal. Wenn “button_press_count“ gleich 2 ist wird auch die zweite LED eingeschaltet. Dann wird Part 2 des Skriptes ausgeführt. Hier wird das Bild gemacht und die Bilder auf den Dateiserver hochgeladen. Am Ende blinken beide LED’s fünfmal. Der “button_press_count“ wird dann wieder auf 0 zurückgesetzt.

```
button_press_count = 0

def button_pressed():
    global button_press_count
    button_press_count += 1

    if button_press_count == 1:
        led_one.on()
        execute_part1()
        flash_led(led_one, 5)
    elif button_press_count == 2:
        led_one.on()
        led_two.on()
        execute_part2()
        flash_both_leds(5)

        button_press_count = 0
```

4.10.8 USB und Case-ID Script

Als erstes werden die globalen Variablen “case_id“ und “file_name“ deklariert. Diese werden verwendet, um den Wert der Case-Id und des Dateinamens zu speichern.
Nun werden mit dem Befehl “subprocess.run“ die Uhubctl Befehle ausgeführt. Zuerst wird der USB-Port des Raspberry Pi’s an den die Webcam angeschlossen ist ausgeschaltet und dann wieder eingeschaltet.
Dann wird der Benutzer mit der “input“ Funktion aufgefordert die Case-ID einzugeben. Dies passiert durch den angeschlossenen Barcode-Scanner, der die Case-ID scannt und dies mit der Eingabetaste bestätigt. Die eingegebene Case-ID ist jetzt als Variable gespeichert.
Mit einer If-Abfrage wird überprüft, ob ein Verzeichnis mit dieser Case-ID bereits existiert. Falls nicht wird ein Ordner mit dem Namen erstellt.
Die aktuelle Zeit wird mit “date_time“ abgerufen und in das Format Jahr-Monat-Tag_StundeMinuteSekunde umgewandelt. Dieser Zeitstempel wird dem Dateinamen vorangestellt, um sicherzustellen, dass jede Datei einen eindeutigen Namen hat.
Der endgültige Dateiname setzt sich also aus Case-ID, dem Zeitstempel und der Endung “.jpg“ zusammen.
Schließlich wird die LED Eins wieder ausgeschaltet.

```
def execute_caseid():
    global case_id
    global file_name
    try:
        subprocess.run(["sudo", "uhubctl", "-l", "1", "-a", "0"])
        subprocess.run(["sudo", "uhubctl", "-l", "1", "-a", "1"])

        case_id = input("Please scan the CASE-ID: ")

        if not os.path.exists(case_id):
            os.makedirs(case_id)

        date_time = datetime.datetime.now().strftime("%Y-%m-%d_%H%M%S")
        file_name = f"{case_id}_{date_time}.jpg"
    finally:
        led_one.off()
```

4.10.9 Foto und Datei Script

Auch hier werden zuerst wieder die globalen Variablen deklariert. Danach wird mit dem “subprocess.run“ die “fswebcam“ Funktion aufgerufen. Diese macht unter den, in 4.6 bereits angegebenen Kamerakonfigurationen das Bild und speichert es lokal auf dem Raspberry Pi ab. Dann wird mit “dest_host“ und “dest_dir“ der Hostname des Dateiservers, und das entsprechende Verzeichnis definiert. Dies sind unter Berücksichtigung des DSGVO abgeänderte Werte.
Mit “subprocess.run“ wird dann über eine SSH-verbindung sichergestellt, dass das Verzeichnis auf dem Dateiserver existiert. Das Bildverzeichnis kann nun mit “scp“ auf den Dateiserver übertragen werden.
Mit einer If-Abfrage wird überprüft, ob das Bildverzeichnis erfolgreich auf den Dateiserver übertragen wurde. Wenn ja, wird das lokale Bildverzeichnis auf dem Raspberry Pi gelöscht und beide LED’s blinken fünfmal. Sollte das Bildverzeichnis nicht übertragen werden können, blinkt nur die zweite LED zehnmal auf. Somit kann sichergestellt werden, dass ein Problem im System erkannt wird und die Fotobox nicht weiter genutzt wird.
Zuletzt werden wieder beide LED’s ausgeschaltet.
Mit der Funktion “pause()“ wird sichergestellt dass das Skript nicht aufhört sondern wieder von vorne beginnt.

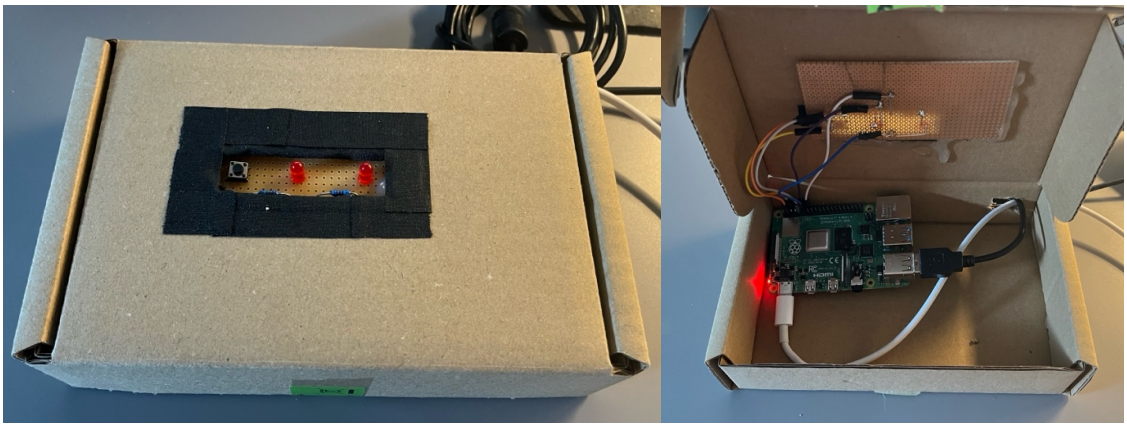
```
def execute_picture():
    global case_id
    global file_name
    try:
        subprocess.run(["fswebcam", "-r", "1920x1080", "-p", "YUYV", "-S", "20", "-D",
"2", "-F", "2", f"/home/admin/camera/{case_id}/{file_name}"])

        dest_host = "enfinitecgmbh@192.168.178.40"
        dest_dir = "/enfinitec/repaircenter/acer/customerdevices/pictures"
        folder_dir = f"/home/admin/camera/{case_id}"
        subprocess.run(["ssh", dest_host, f"mkdir -p {dest_dir}/{case_id}"])
        subprocess.run(["scp", "-r", folder_dir, f"{dest_host}:{dest_dir}"])

        if subprocess.call(["ssh", dest_host, f"test -d {dest_dir}/{case_id}"]) == 0:
            print("Folder transferred successfully, now deleting the local folder.")
            shutil.rmtree(folder_dir)
        else:
            print("Folder transfer failed, local folder not deleted.")
    finally:
        led_one.off()
        led_two.off()
button.when_pressed = button_pressed
pause()
```

4.10.10 Das Python-Skript ausführen

Mit **python camera.py** kann nun das Python-Skript ausgeführt werden. Da der Raspberry Pi aber mit keiner Tastatur ausgestattet ist lassen wir das Skript direkt beim Hochfahren des Raspberry Pi's starten. Dies können wir festlegen indem wir die rc.local Datei des Raspberry Pi's mit „sudo nano" bearbeiten. Hier schreiben wir nun am Ende der Datei, unser Python-Skript und das Verzeichnis, in dem es liegt, hinein:
"sudo python /home/admin/camera/camera.py &"
Das & Zeichen stellt sicher, dass das Skript in einem separaten Prozess läuft und nicht im normalen Bootprozess des Raspberry Pi's.
Nun starten wir den Raspberry Pi neu und unser Python-Skript startet automatisch.

4.11 Die Fotobox

Um die Bauteile und den Raspberry Pi kompakt und mobil zu halten, habe ich mich dazu entschieden alles in eine Box zu installieren. Unter Berücksichtigung der Umwelt und Nachhaltigkeit habe ich mich für eine einfache Box aus Pappe, die sonst im Müll gelandet wäre, entschieden. Dies beeinträchtigt die Funktion in keiner Weise und birgt keinerlei Nachteile im Vergleich zu einem Gehäuse, das beispielsweise 3D-gedruckt wurde.



5 Fazit

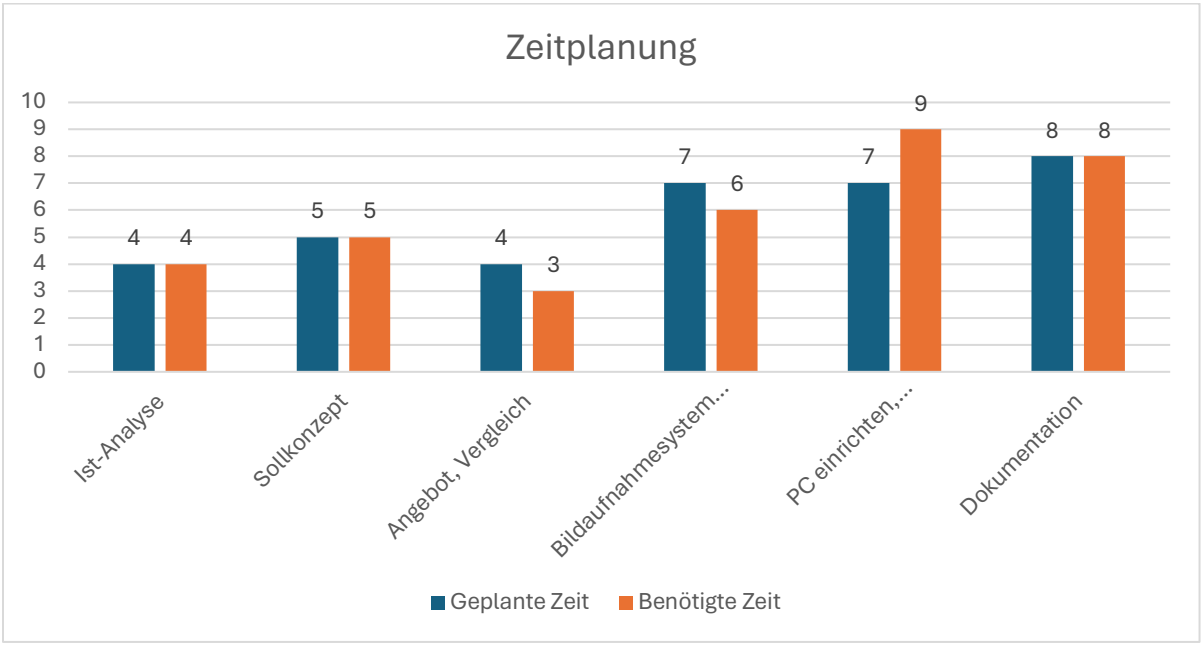
5.1 Fazit des Projektes

Insgesamt war das Projekt ein voller Erfolg. Die Fotobox funktioniert ohne Probleme und die Bedienung ist einfach zu verstehen. Meine Projektplanung konnte sehr gut eingehalten werden und es gab keine schwerwiegenden Probleme. Es konnte insgesamt alles in der vorgeschriebenen Zeit durchgeführt werden, auch wenn es, wie in **5.3** beschrieben, innerhalb der einzelnen Punkte kleine Abweichungen gab.

5.2 Persönliches Fazit

Mir hat das Projekt sehr viel Spaß gemacht, da ich mich privat auch gerne mit Raspberry Pi's beschäftige. Das Programmieren des Skriptes in Python war manchmal kompliziert und ich musste einiges recherchieren, habe dadurch aber auch vieles gelernt.

5.3 Zeitplanung



6 Glossar

Add-on 13
Ein Add-on, auch bekannt als Erweiterung oder Plugin, bezieht sich auf eine zusätzliche Softwarekomponente, die in eine bestehende Anwendung integriert wird, um deren Funktionalität zu erweitern oder anzupassen. Im Kontext von Visual Studio Code sind Add-ons kleine Softwarepakete, die spezifische Funktionen, Tools oder Unterstützung für verschiedene Programmiersprachen, Frameworks oder Arbeitsabläufe bereitstellen.

Arbeitsspeicher 9
Der Arbeitsspeicher ist ein Bereich im Computer, der Daten temporär speichert, damit sie schnell von der CPU abgerufen werden können.

Breadboards 12
Ein Breadboard ist ein Hilfsmittel in der Elektronik für die schnelle Prototypenerstellung von Schaltungen. Es ermöglicht das Verbinden elektronischer Bauteile ohne Löten durch elektrisch verbundene Löcher. Ideal zum Entwerfen, Ändern und Testen von Schaltungen vor dem Verdrahten auf Leiterplatten.

Contact Center Suite 6
Die Contact Center Suite ist ein hauseigenes Programm in der die verschiedenen Kundengeräte verwaltet werden. Hier werden Kostenvoranschläge bearbeitet, Ersatzteile bestellt und Kundendaten ausgelesen.

E-Mobility 3
Bezieht sich auf elektronisch angetriebene Roller, Segways und Karts.

Ethernet 4
Ethernet ist eine Technologie für die Verkabelung von Computernetzwerken, die häufig für lokale Netzwerke (LANs) verwendet wird.

for-schleife 14
For-Schleife ist eine Struktur in der Programmierung, die dazu dient, eine bestimmte Anweisung oder Gruppe von Anweisungen wiederholt auszuführen. Man verwendet sie, um eine definierte Anzahl von Wiederholungen durchzuführen.

fswebcam 10
fswebcam ist ein Befehlszeilenprogramm unter Linux, welches verwendet wird, um Bilder von einer angeschlossenen Webcam aufzunehmen. Mit fswebcam können Benutzer schnell und einfach Fotos von einer unterstützen Webcam aufnehmen, indem sie Parameter wie Bildauflösung, Dateiformat, Ausgabequalität und Aufnahmeintervall festlegen.

GPIO-Pins 7
GPIO-Pins (General Purpose Input/Output-Pins) sind physische Anschlüsse auf dem Raspberry Pi, die es ermöglichen, digitale Signale zu senden oder zu empfangen. Diese Pins können sowohl als Eingang(Input) als auch als Ausgang(Output) konfiguriert werden, was dem Raspberry Pi die Interaktion mit einer Vielzahl von externen Geräten und Sensoren ermöglicht.

Headless 10
Headless bezieht sich auf den Betrieb eines Raspberry Pi ohne angeschlossenen Monitor, Tastatur oder Maus. Bei einem headless Raspberry Pi wird das Gerät über das Netzwerk ferngesteuert, normalerweise über SSH oder andere Remote Zugriffsmethoden. Dies ermöglicht es dem Benutzer, den Raspberry Pi zu konfigurieren, Programme auszuführen und auf Dateien zuzugreifen, ohne physisch mit dem Gerät verbunden zu sein.

Home Verzeichnis 12
Das Home-Verzeichnis ist der persönliche Bereich eines Benutzers auf einem Unix-basierten Betriebssystem wir Linux oder macOS. Jeder Benutzer hat sein eigenes Home-Verzeichnes das mit dem Benutzernamen identifiziert wird.

7 Quellen

https://www-users.york.ac.uk/~mjf5/shed_cam/src/USB%20webcam.html

https://www.youtube.com/watch?v=_uVaZalaSbI

https://github.com/mvp/uhubctl?tab=readme-ov-file#raspberry-pi-4b

https://www.circuitbasics.com/how-to-write-and-run-a-shell-script-on-the-raspberry-pi/

https://alvinalexander.com/linux-unix/how-use-scp-without-password-backups-copy/

https://www.berrybase.de/raspberry-pi/raspberry-pi-computer/boards/

https://stackoverflow.com/questions/11304895/how-do-i-copy-a-folder-from-remote-to-local
using-scp

https://www.enablegeek.com/tutorial/install-python-on-a-raspberry-pi-step-by-step-guide/

https://www.baeldung.com/linux/python-run-bash-command

https://www.python.org/downloads/

https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/

https://www.tinkercad.com

https://www.reichelt.de/