

Bachelor Thesis

Planung und Programmierung
einer physischen
Zwischeninstanz im
Druckprotokoll und
Verarbeitung der Bilddateien
mit Sensordaten

Dresden
23.06.2022

Bachelor Thesis

Planung und Programmierung einer physischen Zwischeninstanz im Druckprotokoll und Verarbeitung der Bilddateien mit Sensordaten

Von

Simon Awißus

Zur Erlangung des Grades

Bachelor of Science

In Angewandter Informatik

an der Hochschule Konstanz – Technik, Wirtschaft und Gestaltung

Matrikelnummer: 298744

Abgabetermin: 30.06.2022

Betreuer: **Prof. Dr. Rebekka Axthelm**

Zweit-Betreuer: **Dipl. Inform. (FH) Christian Böhm**

Eidesstattliche Erklärung

Hiermit erkläre ich, *Simon Awißus*, geboren am 15.01.1999 in Berlin,

- (1) dass ich meine Bachelorarbeit mit dem Titel
**Planung und Programmierung einer physischen
Zwischeninstanz im Druckprotokoll und Verarbeitung
der Bilddateien mit Sensordaten**
selbstständig und ohne fremde Hilfe angefertigt habe und
keine anderen als die angeführten Hilfen benutzt habe;
- (2) dass ich die Übernahme wörtlicher Zitate, von Tabellen,
Zeichnungen, Bildern und Programmen aus der Literatur
oder anderen Quellen (Internet) sowie die Verwendung der
Gedanken anderer Autoren an den entsprechenden Stellen
innerhalb der Arbeit gekennzeichnet habe
- (3) dass die eingereichten Abgabe-Exemplare in Papierform und
um PDF-Format vollständig übereinstimmen.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen
haben wird.

Dresden, 30.06.2022

(Unterschrift)

Inhaltsverzeichnis

1	Abstract	5
2	Einleitung.....	6
3	Problemstellung	6
4	Funktionelle Spezifikation	7
4.1	Einrichtung	7
4.2	Handhabung	7
4.3	Visuelle Erscheinung	8
5	Theoretischer Hintergrund	9
5.1	Raspberry Pi	10
5.2	Projektorboard	11
5.3	Distanzmessung mit Abstandssensoren	11
5.4	Drehimpulsgeber	12
5.5	Webserver.....	12
5.6	Bildformate	12
5.6.1	JPEG.....	12
5.6.2	PNG	12
5.6.3	PDF	13
5.7	Bibliotheken zur Bildmanipulation	13
5.8	IPP-Protokoll (Internet Printing Protocol).....	14
5.9	I2C-Protokoll (Inter-Integrated Circuit)	14
5.10	Transferpapierdrucker	14
6	Technische Spezifikation	15
6.1	Analyse.....	15
6.1.1	Auswahl der Programmiersprache	15
6.1.2	Auswahl der Bibliothek	16
6.1.3	Berechnung von Breite und Höhe des finalen Bildes.....	16
6.1.4	Frequenz des Abstandssensors	17
6.1.5	Auswahl des internen Datentyps.....	18
6.2	Konzept	18
6.2.1	Hardware.....	18
6.2.2	Software	19
6.2.3	Gehäuse	20

7	Dokumentation der Implementierung	21
7.1	Emulieren eines Druckers	21
7.1.1	Evaluation.....	21
7.1.2	Fazit.....	21
7.2	Verdrahtung der Bauteile	22
7.2.1	Abstandssensor	22
7.2.2	Projektorplatine.....	22
7.2.3	Drehimpulsgeber	24
7.2.4	Fazit.....	24
7.3	Oberfläche.....	25
7.3.1	Website	25
7.3.2	Webserver	27
7.3.3	Fazit.....	27
7.4	Firmware.....	27
7.4.1	Darstellung der Dateien	28
7.4.2	Optionen zur Bildoptimierung.....	28
7.4.3	Empfangen und Glätten der Abstands Daten.....	29
7.4.4	Verrechnen der Bilder mit den Sensor Daten	29
7.4.5	Linienfinder bei Handfotos	31
7.4.6	Senden des Druckauftrags.....	32
7.5	Gehäuse	33
8	Evaluation	34
8.1	Testreihe	34
8.1.1	Effizienz der Filter	34
8.1.2	Projektion auf verschiedenen Körperteilen	34
8.1.3	Größenberechnung des finalen Drucks.....	35
8.2	Gütemaß	35
8.3	Kostenberechnung	36
9	Zusammenfassung	36
10	Ausblick.....	37
10.1	Kalibrierung.....	37
10.2	Mögliche Verbesserungen	37
10.3	Persönliche Meinung.....	37
11	Quellennachweis.....	38
12	Abbildungsverzeichnis	39
13	Tabellenverzeichnis	40

1 Abstract

Die vorliegende Bachelor-Arbeit beschäftigt sich mit der Gestaltung eines Prototyps für ein Tätowier-Studio. Hierfür müssen verschiedene Sensor Daten auf einem Einplatinencomputer verwertet und aufbereitet werden. Die zu Grunde liegenden Prozesse ragen in die Bereiche der Bildmanipulation, der Front- und Backend Entwicklung, sowie des Druckprotokolls. Wichtig ist, die komplexen Vorgänge hinter dem Gehäuse des Prototyps für den User zu kaschieren und ihm eine möglichst nutzerfreundliche und simple Oberfläche für den Prototypen anzubieten. Im Rahmen dieser Ausarbeitung werden die funktionellen und technischen Abläufe aufgrund der Wünsche des Kunden erarbeitet und umgesetzt. Die Fertigstellung des Prototyps zu einem marktfähigen Produkt ist jedoch kein Teil dieser Arbeit. Dennoch soll in einem späteren Kapitel dieser Arbeit auf die nötigen Schritte zum kommerziellen Produkt eingegangen werden, um offene Fragen und zu klärende Problemstellungen des Prototyps hervorzuheben.

2 Einleitung

Mein Interesse daran, Neues zu erfinden und zu bauen ragt bis in meine Kindheit. Im Jugendalter hatte ich erstmals Kontakt mit den E/A-Boards der Firma Arduino, die eine Welt des Programmierens für mich eröffneten. Über die Jahre setzte ich viele kleinere Projekte um, welche ich dann sogar ab und zu im Alltag einsetzen konnte. Im Wintersemester 2021 erzählte mir eine befreundete Tätowiererin von einer Idee, die ihr während ihres Arbeitsalltages im Studio kam. Ich war sofort gepackt und setzte mir in den Kopf die Idee in die Tat umzusetzen.

3 Problemstellung

Ein Tätowier- Studio arbeitet meist nach dem folgenden Schema:

Ein Bild, welches dem Kunden tätowiert werden soll, befindet sich auf einem Computer, meistens ein Tablet, im Studio. Der Tätowierer druckt dieses Bild zunächst aus, damit eine Entscheidung über Größe und Position des Bildes auf dem Körper des Kunden getroffen werden kann. Das Bild wird also an die betreffende Körperstelle gehalten. Um die Größe zu variieren, muss das Bild öfter in verschiedenen Größen gedruckt werden. Schließlich erfolgt ein endgültiger Ausdruck auf Thermotransferpapier, von welchem dann der Umriss des Bildes zunächst grob auf den Körper übertragen wird. Erst dann kann das Tätowieren beginnen.

Dieses Vorgehen ist zeitaufwendig und wenig effektiv. Deshalb wurde die Idee entwickelt, die Tätowier-Vorlage direkt mittels eines kleinen Beamers auf Raspberry-Basis auf den Körper des Kunden zu projizieren.

Der Prototyp dieses Beamers soll die Fähigkeit besitzen, die Bilder direkt in der Größe zu variieren, in dem zum Beispiel der Abstand zwischen Beamer und Körper verändert wird, oder durch eine entsprechende Steuerung am Gerät.

Nach Auslösen des Druckvorganges ermittelt der Prototyp die tatsächliche Größe des Bildes mit Hilfe einer Abstandsbestimmung zwischen Beamer und Körper und löst den Druckvorgang für den Transferpapierdruck aus.

Der Tätowierer spart durch diesen Vorgang Zeit und Druckpapier, der Kunde kann das geplante Ergebnis vorab auf seinem Körper begutachten und entscheiden.

4 Funktionelle Spezifikation

Die funktionelle Spezifikation wird in enger Zusammenarbeit mit der Tätowiererin ausgearbeitet, um ein Konzept für den Prototypen zu erstellen. Im Gespräch werden Abläufe wie die Einrichtung des Gerätes sowie die Handhabung und Optik des Gerätes thematisiert. Durch diesen Vorgang lassen sich spätere Missverständnisse in der Funktionsweise ausschließen. Als Entwickler des Prototyps ist es enorm wichtig, während der Implementierung, den User und seine Herangehensweise zu berücksichtigen. Nur so kann der Prototyp eine echte Hilfe in Tätowier-Studios werden.

4.1 Einrichtung

Vor der Benutzung des Gerätes müssen aus technischer Sicht einige Einrichtungsschritte vorgenommen werden.

Der erste wichtige Einrichtungsschritt ist die Stromversorgung, welche entweder über Akku oder per Kabel erfolgen kann. Im Falle eines Akkus, welcher deutlich praktischer für den alltäglichen Gebrauch wäre, muss der Prototyp vor der Verwendung aufgeladen werden. Anschließend wird der Prototyp mit dem lokalen WLAN-Netzwerk verbunden.

Sind diese Schritte erledigt, kann der User auf die Weboberfläche des Prototyps zugreifen, um das Gerät anzusteuern. Dort können weitere Einstellungen zur Verbesserung der Bilddateien vorgenommen werden.

4.2 Handhabung

Der User betätigt den Upload Knopf auf der Website des Beamers. Daraufhin wird ihm das Dateisystem des Gerätes angezeigt. Hier sucht der User das zu verarbeitende Bild und bestätigt die Auswahl.

Alternativ kann der User das Bild auch in einem freigegebenen Netzwerkpfad direkt im Zielordner auf dem Server ablegen.

Der User kann außerdem in der Bildansicht auf seinem Tablet das Druckprotokoll aufrufen. Dort wird ihm der Prototyp als Drucker angezeigt. Drückt der User auf Drucken, wird die Datei an den Server geleitet.

Sobald das Bild bei dem Prototyp angekommen ist, zeigt dieser das Bild auf der Projektorplatine an. Jetzt nimmt der User den Prototypen zur Hand und hält ihn in einem angebrachten Abstand auf die Stelle, wo das Tattoo geplant ist. Während des Vorgangs signalisiert der Prototyp, ob der gehaltene Abstand im messbaren Bereich ist. Der User hat zwei Möglichkeiten auf die Größe des projizierten Bildes Einfluss zu nehmen. Das ist der Abstand zur Haut des Kunden und das Betätigen eines kleinen Rades am Gerät. Sobald der User mit der Größe zufrieden ist, wird der Okay- Knopf betätigt. Nach einer kurzen Verarbeitungszeit entnimmt der User dem Drucker den fertigen Transferpapierdruck. Dieser hat die exakte Größe des auf die Haut projizierten Bildes. Den Druck kann er nun für die weitere Arbeit verwenden. Um ein weiteres Bild zu verarbeiten, wählt der User auf der Weboberfläche einfach ein neues Bild aus.

4.3 Visuelle Erscheinung

Bezüglich der Optik wurden weniger präzise Aussagen getätigt. Grundsätzlich sollten die Regler und Knöpfe des Prototyps komfortabel erreichbar sein. Die Tätowiererin wünscht sich zudem, dass der Drehregler oben auf dem Gehäuse des Prototyps ist. Außerdem wäre es praktisch das Gerät abstellen zu können, um genaue Messungen ohne Ungenauigkeiten durchführen zu können. Wichtig ist außerdem, dass die Schärfeneinstellung der Linse von außen möglich ist. Grundsätzlich sollte das Gehäuse einen modernen Look bekommen.

5 Theoretischer Hintergrund

Um ein effektives Konzept für einen Prototypen entwickeln zu können, sind einige Grundlagen über die technischen Bestandteile des Projektes von Nöten. Da außerdem eine Bildverarbeitungssoftware entworfen werden muss, sollte sich außerdem über die richtige Programmiersprache und verwendbare Bibliotheken Gedanken gemacht werden. Die einzelnen Bestandteile sollten sich im unteren Kostensektor bewegen, damit das fertig entwickelte Produkt zu einem bezahlbaren Preis verkauft werden kann. Die erste Grundlage für einen intelligenten Beamer, ist der Prozessor. Für diesen Teil des Prototyps bietet der Markt eine reichliche Auswahl an Mikrocontrollern. Eine Möglichkeit wäre es, sich an einem der Mikrocontroller der Marke Arduino zu versuchen. Diese Bauteile sind dazu in der Lage, analoge und digitale Eingangssignale zu lesen und zu verarbeiten (Banzi, Shiloh, 2022). Dazu wird das selbst geschriebene Programm via USB auf den internen Speicher der Platine gespielt. Dort läuft es dann als Endlosschleife, bis der Arduino vom Stromnetz gelöst wird. Ein Problem der Arduino Mikrocontroller ist allerdings, dass die Geräte nicht gut für visuelle Darstellungen zu gebrauchen sind. Zwar können die Geräte als Netzwerkakteure Daten abrufen und verarbeiten, aber die Darstellung der Bilder auf einem Projektor ist so gut wie unmöglich. Diese Tatsache schließt die Arduino Mikrocontroller für den Prototypen aus.

5.1 Raspberry Pi

Die Idee eines Einplatinencomputers für Entwicklungs- und Lehrzwecke stammt bereits aus den 80er Jahren. Damals waren die Prozessoren jedoch noch sehr teuer, weshalb die Platinen meist nur in Produktsystemen in der Industrie verwendet wurden. Der weitreichende Einsatz der Platinen für Lehrzwecke war lange undenkbar.

Im Jahr 2006 entwickelte die britische Firma Raspberry Pi Foundation aber einen neuartigen Einplatinencomputer, der jungen Menschen das Programmieren und Experimentieren rund um das Thema Informatik ermöglichen sollte (Ortmeyer, 2014). Der Computer hat ungefähr die Größe einer Kreditkarte. Ein Raspberry Pi kostet im Durchschnitt zwischen 20 bis 100 Euro. Durch die Chipkrise des vergangenen Jahres sind die Preise für die neueren Modelle angestiegen. Der Preis für einen Raspberry Pi 4 beträgt inzwischen mehr als 100 Euro. (o.D.[a], 2022)



Abbildung 1: Raspberry Pi 3 - Platine

Der Einplatinencomputer eignet sich, trotz seiner simplen Architektur auch für den Produktiveinsatz. Das Betriebssystem, für gewöhnlich eine Linux Distribution, befindet sich auf einer SD-Karte. Auf den Grafikprozessor wurde bei der Entwicklung nicht viel Wert gelegt. Die leistungsstärkste GPU der Raspberry-Reihe, verbaut im Raspberry Pi 4, schafft gerade einmal 500Mhz. Im Vergleich dazu, ein einzelner Kern des iPhone 13 Pro Max verfügt über 3,2 GHz. (La Rocco, 2021). Der Raspberry Pi 3 Modell B wurde bereits 2016 entwickelt und zeichnet sich durch seine WLAN und Bluetooth Fähigkeit aus. Die eingebaute CPU ist eine auch in Smartphones verwendete 64-bit-Armv8-Architektur. Die CPU schafft eine Taktfrequenz von bis zu 1200MHz (Raspberry Pi Model Specifications, 2014). Damit sollte die geplante Software des Prototyps problemlos laufen.

Seine schmale Größe und der niedrige Preis des Raspberry Pi 3 machen ihn zur perfekten Steuereinheit für den Prototypen.

5.2 Projektorboard

Ein Projektorboard ist ein Beamer im Kleinstformat. Über einige Datenpins können die Videodaten an die Platine des Projektorboards geschickt werden. Das Board gibt die Videodaten über den am vorderen Ende angebrachten Projektor aus. An der Linse befindet sich außerdem ein kleiner Hebel, mit dem sich die Linse auf verschiedene Distanzen scharfstellen lässt. Da die Projektorplatinen im Kreditkartenformat oft keine Kühlung verbaut haben, erhitzen sich Projektor und Platine schnell. Diese Tatsache ist für die spätere Konstruktion des Gehäuses relevant.

5.3 Distanzmessung mit Abstandssensoren

Grundsätzlich gibt es einige Ansätze, um den Abstand zwischen Sensor und einem Objekt zu messen. Die Technologie findet in vielen industriellen Bereichen Anwendung und wurde so über die Jahre immer weiterentwickelt. Grundsätzlich spaltet sich die moderne Distanzmessung in aktive und passive Berechnung auf. Ein aktiver Ansatz ist es immer dann, wenn für die Berechnung Licht „ausgesendet“ wird. Eine Version eines aktiven Ansatzes ist beispielsweise die Triangulation, bei der ein Laser und eine Kamera benötigt werden. Der Laser wird in einem genau kalibrierten Winkel auf das Objekt gestrahlt. Anhand der Differenz des zurückgeworfenen Lichtes lässt sich dann der Abstand zum Objekt berechnen. Ein weiterer aktiver Ansatz ist die Shape-from-X-Methode, bei der die Verzerrung eines projizierten Musters auf eine Oberfläche erkannt und in einen Abstandswert umgerechnet wird.

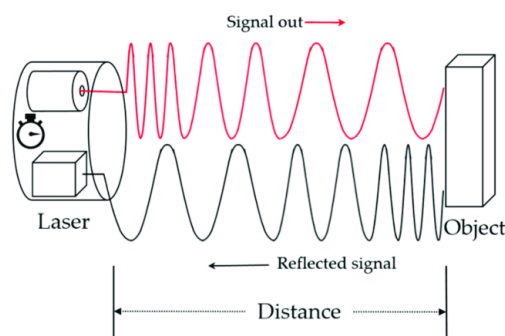


Abbildung 2: Arbeitsweise eines Time-of-Flight-Sensors

Der am weitesten verbreitete und älteste aktive Ansatz ist die Time-of-Flight Methode. Hierbei misst man die Zeit, die ein Laser braucht, um auf das Objekt zu treffen und in die Kamera zurück reflektiert zu werden. Sensoren dieser Art sind sowohl präzise als auch kostengünstig. Die Distanz d berechnet sich mit der Formel:

$$d = \frac{t_1 - t_0}{2c}$$

Der Zweig der passiven Triangulation hingegen findet meist im Bereich der Computer-Vision Anwendung. Für die passive Abstandsberechnung werden immer mindestens zwei Bilder des Objektes benötigt. Die Differenz der zwei Kameraaufnahmen gibt dann Auskunft über die Entfernung zum aufgenommenen Objekt. So kann man beispielsweise eine Kamera langsam um ein Objekt bewegen und über die Stereo-Vision-Algorithmen kann eine virtuelle Punktwolke für das Objekt generiert werden.

5.4 Drehimpulsgeber

Ein Drehimpulsgeber, oder auch Rotary Encoder, ist ein Sensor zur Berechnung eines Drehwinkels. Am häufigsten findet der Sensor heute in Computermäusen Anwendung. Diese Art von Sensor liefert zwei digitale Ausgangssignale aus, Signal A und Signal B. Jedes der Signale kennt nur entweder den Zustand HIGH oder LOW. Durch die Drehung des Sensors erhalten die Signale nacheinander abwechselnd den Zustand HIGH und LOW. Der Rhythmus der Zustände gibt dann Auskunft über die Drehbewegung des Sensors. Für die Umrechnung der digitalen Signale in die einzelnen Größenstufen der Bilder wird später ein kleiner Codeabschnitt verantwortlich sein, der auf eingehende Signale des Drehimpulsgebers achtet.

5.5 Webserver

Ein Webserver ist ein Computer, der von anderen Geräten mithilfe einer URL angesprochen werden kann. Über den Router können andere Geräte, die Clients, Anfragen an den Server schicken. So können beispielsweise HTML Seiten im Speichersystem des Servers abgerufen werden. Diese werden dann im Browser des Clients visualisiert. Der Server kann auch Daten generieren und für die Clients bereitstellen oder visuell aufbereiten. Ein häufig auf Linux verwendeter Server ist der Apache Webserver. Dieser Webserver wurde bereits 1995 veröffentlicht und hat sich trotz großer Konkurrenz als solide Lösung etabliert. (Behlendorf, 2012)

5.6 Bildformate

5.6.1 JPEG

Das JPEG-Format ist eine komprimierte Bilddatei. Dabei unterscheidet man zwischen verlustbehafteter und verlustfreier Kompression. Die eigentlichen komprimierten Bilddaten werden dann als Grafikformat JFIF, SPIFF oder JNG abgelegt. Die Komprimierung erfolgt in einigen Verarbeitungsschritten, die dann bei der Dekomprimierung in umgekehrter Reihenfolge angewendet werden. Die Komprimierungsschritte können manuell angepasst werden. JPEGs oder auch JPGs sind besonders oft im Netz anzutreffen, da der Datentyp die auf Websites angezeigten Bilder leicht in verschiedene Größen umrechnen kann (Miano, 1999).

5.6.2 PNG

Das PNG-Format ist ein Rastergrafikformat. Es ermöglicht die verlustfreie Datenkompression. Besonders ist bei PNG-Bildern der zusätzliche Alphakanal, welcher zum Speichern von Transparenzwerten dient. Das sorgt auch dafür, dass ein PNG-Bild meist etwas größer ist als seine Entsprechung im JPEG-Datenformat. Die Komprimierung des Datenformats setzt sich aus einigen, teils optionalen Schritten zusammen. Als Erstes werden die Bereiche des Bildes nach Ähnlichkeit kategorisiert, die Dekorrelation. Anschließend kann das Bild mittels Substitution komprimiert werden (MiQano, 1999).

Diese Komprimierung wird in den Bibliotheken auch zur Vergrößerung und Verkleinerung des Bildes verwendet. Der Datenheader eines PNG-Bildes enthält einige Pflichtblöcke IHDR, IDAT, PLTE, IEND. Darauf folgen die „Chunks“, die elementaren Datenstrukturen der PNG Datei (Miano, 1999).

5.6.3 PDF

Die PDF-Datei wurde für die einfache Migration von Schriftstücken auf verschiedene Geräte und Plattformen entwickelt. Das von Adobe entwickelte Datenformat lässt sich auf die Funktionsweise des Postscriptformates zurückführen, welches auch im Druckprotokoll eine große Rolle spielt. Durch die Entwicklung der PDF konnten viele Barrieren der Dokumentenmigration behoben werden. Eine PDF kann Text, Grafik aber auch Bilder enthalten (Knijff, 2009). Inzwischen können fast alle Grafikprogramme PDF-Dateien ausgeben. Die nachträgliche Bearbeitung der Dateien ist allerdings deutlich komplizierter als in den vorher genannten Datenformaten. Es sind spezielle Programme notwendig, um beispielsweise Seiten einer PDF zu entfernen oder zu zerschneiden. Auch das Ändern der Größe stellt eine deutlich größere Hürde dar.

5.7 Bibliotheken zur Bildmanipulation

Für die Verrechnung der Bilddatei mit den Sensordaten kann auf eine reiche Anzahl von Bibliotheken zurückgegriffen werden. Die meisten Programmiersprachen bieten in dieser Hinsicht anwendbare Methoden an.

Eine der bekanntesten und am weitesten entwickelten Bibliotheken zur Pixelmanipulation ist *OpenCV*. Diese mächtige Library bietet sogar in den Bereichen der Gesichtserkennung oder auch des Deep-Learning Funktionen an. Besonders interessant für das Projekt sind aber die Möglichkeiten der Bildverarbeitung und das schnelle Anwenden von simplen Filtern, zum Beispiel der Gauß-Filter oder der Sobel-Filter. Die Bibliothek ist sowohl in Python, C und C++ benutzbar (Brahmbhatt, 2013).

Eine weitere interessante Bibliothek ist *Scikit-image*, eine Bibliothek, die viele Algorithmen für Wissenschaft, Industrie und Bildung bereithält. Die umfangreiche Bibliothek bietet Segmentierung, Transformation der Bilder und auch Filterung und Analyse der Bilder. Die Library ausschließlich für die Sprache Python konzipiert. (Van der Walt, 2014)

Eine weitere für Python ausgelegte Bibliothek ist die Python Imaging Library, kurz *PIL*. Diese externe Python Bibliothek ist ursprünglich für Python 2 entwickelt worden. 2011 wurde das Projekt jedoch unter dem Namen *Pillow* geklont und für Python 3 angepasst. Die Library ermöglicht Manipulationen auf Pixelebene, Masking und Transparenzbearbeitung und stellt viele Filter bereit, die für den Prototypen von Vorteil sein können (Clark 2019).

Jede der aufgeführten Bibliotheken kann mit allen gängigen Bildformaten arbeiten. Die Manipulation von PDF-Dateien jedoch ist für keinen der aufgeführten Kandidaten möglich. Die einzige Möglichkeit eine PDF-Datei zu bearbeiten ist es, diese in ein Bildformat umzuwandeln. Eine Python-Bibliothek mit dem Namen *pdf2image* ist dazu in der Lage PDF-Dateien mit wenigen Zeilen Code in ein Bild umzuwandeln. Für C++ gibt es eine ähnliche Bibliothek mit dem Namen *ImageMagick*.

5.8 IPP-Protokoll (Internet Printing Protocol)

Das IPP-Protokoll ermöglicht das Senden und Empfangen von Druckaufträgen über das Internet oder lokale Netzwerke. Es vereint alle notwendigen Mechanismen wie Authentifizierung, Verschlüsselung und Statusmeldungen gemeinsam in einem einzigen Protokoll. So kann beispielsweise ein Tablet die Drucker im Netzwerk erkennen und Druckaufträge an diese schicken. Das IPP-Protokoll basiert grundlegend auf der TCP/IP Technologie von HTTP. Es gibt im IPP-Protokoll grundsätzlich zwei Arten von Objekten: Jobobjekte und Printerobjekte. Über verschiedene Verzeichnisse können diese Objekte abgerufen werden. Für bequemes Drucken auch aus Linux- und Unix- Systemen verwendet man am besten das Common Unix Printing System (CUPS) in Verbindung mit dem IPP-Protokoll (o.D.[c] 2020).

5.9 I2C-Protokoll (Inter-Integrated Circuit)

Das I2C-Kommunikationsprotokoll verwendet zwei bidirektionale Leitungen, die Serial Data Line (SDA) und die Serial Clock Line (SCL). Man unterscheidet grundsätzlich zwischen Master- und Slave-Geräten. Ein Master kann dabei mit mehreren Slave-Geräten verbunden werden, da jeder Slave eine eindeutige Adresse besitzt. Die Kommunikation beginnt mit der Initialisierung durch den Master. Er signalisiert allen wartenden Geräten auf der Datenleitung nach Anweisungen zu suchen. Anschließend beginnt die I2C-Kommunikation, indem die Adresse des gesuchten Slave-Gerätes mit einer Lese- und Schreib-Flag auf den Bus geschrieben wird. Sobald der Slave mit einem Bestätigungssignal antwortet, beginnt die Datenübertragung auf der SDA Leitung. Die SCL Leitung dient zur Übertragung der Master-Clock, also der Geschwindigkeit, in der die Daten übertragen werden. Die Übertragung endet nach einem Stoppsignal des Masters (Kalinsky 2009).

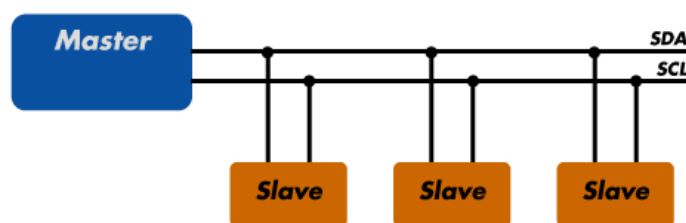


Abbildung 3: Master-Slave System des I2C-Protokolls (o.D[d])

5.10 Transferpapierdrucker

Diese speziell für Tätowier-Studios konzipierten Drucker arbeiten mit Transferpapier, statt dem normalen Druckpapier. Dieses Papier ermöglicht die Übertragung des Designs auf den Körper des zu tätowierenden Kunden. So entsteht ein temporäres Tattoo auf der Haut des Kunden. Anhand dieses temporären Tattoos kann man abschätzen, wie das fertige Tattoo später aussehen wird und es dient dem Tätowierer als Vorlage für seine Arbeit. Transferpapierdrucker, auch Stencil-Drucker genannt, arbeiten am besten mit Bildern deren Linien einen extremen Kontrast zum Hintergrund haben. Die besten Drucke gelingen mit Schwarz-Weiß-Bildern.

6 Technische Spezifikation

6.1 Analyse

Aus der funktionellen Spezifikation lässt sich ein Zustandsdiagramm für den Prototypen ableiten. Die Übergänge zwischen den Zuständen werden durch Eingaben des Users verursacht. Nach erstem Start des Systems gelangt der Prototyp in den Konfigurierungs-Modus. Hier wird das Gerät kalibriert und konfiguriert. Anschließend gelangt der Prototyp in den Idle-Modus, in dem das Gerät auf Aufträge wartet. Sobald der User ein Bild hochlädt, zeigt der Prototyp im aktiven Modus das Bild an und skaliert es wie vom User gewünscht. Tätigt der User den Okay-Knopf, druckt der Prototyp das Bild auf dem vom User ausgewählten Drucker.

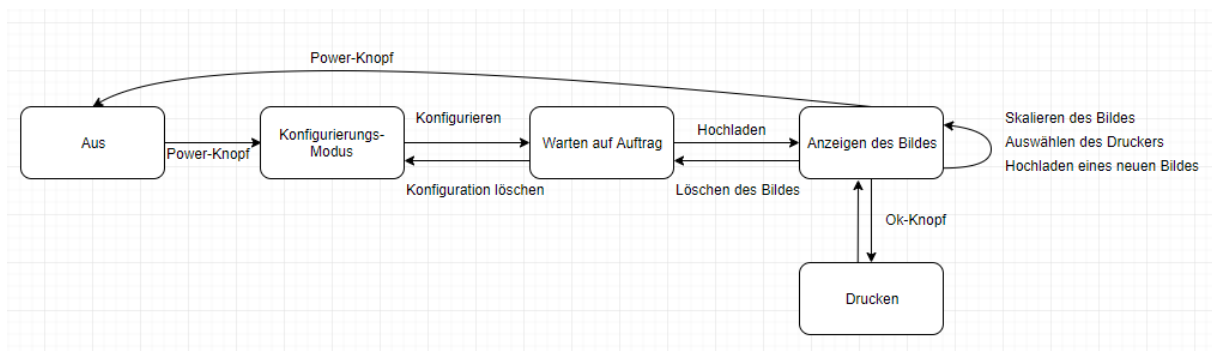


Abbildung 4: Zustandsdiagramm des Prototyps

6.1.1 Auswahl der Programmiersprache

Für die Programmierung der Oberfläche auf dem Apache Webserver wird HTML und CSS verwendet. Da die Website keine komplexen Bestandteile hat, reicht hier Bootstrap als Library für HTML aus. Die Bilddateien können per AJAX verlustfrei an den Server gesendet werden. Dieser speichert die Dateien für die weitere Verwertung.

Die Software für die Sensorik und die Berechnung der Bilder stellt eine kniffligere Frage dar. Hier spielt die Performance der Programmiersprache eine Rolle, außerdem sollten die Libraries der Sprache die Handhabung mit Bilddaten möglichst stark vereinfachen. Auch die Verwendung der Sensorik sollte keine Hürde darstellen.

Die in der Recherche gefundenen Bibliotheken zur Bildmanipulation sind ausnahmslos für die Benutzung mit der Sprache Python ausgelegt. *OpenCV* ist die einzige der Bibliotheken, die auch mit C funktionieren würde. Python ist eine sehr gut dokumentierte, moderne Programmiersprache, die sich für die Verwertung von Daten ausgezeichnet eignet. Alle benötigten Funktionen werden mit der Sprache abgedeckt. Außerdem eignet sich Python hervorragend, um die Sensorik auf dem Raspberry Pi auszulesen.

Für die Programmierung der Firmware des Prototyps, einschließlich der Anbindung des Abstandssensors, des Drehimpulsgebers und der Knöpfe, fiel die Entscheidung auf die Programmiersprache Python.

Sollte sich im späteren Verlauf ein Performanceproblem ergeben, muss überlegt werden, die Firmware des Gerätes alternativ in C zu implementieren. C ist eine performantere Sprache als Python, da sie viel näher am Betriebssystem liegt, welches auch mit C programmiert ist. Zusätzlich erlaubt C im Gegensatz zu Python echtes Multithreading. Pythons Threading ist virtuell und ermöglicht damit nicht die optimale Nutzung der Ressourcen. Das in C enthaltene Threading-Verfahren dagegen lässt eine optimale Aufteilung der Prozesse auf die CPU-Kerne zu.

6.1.2 Auswahl der Bibliothek

Um die Größe der Bilder wie gewünscht zu verändern, kommen alle oben genannten Bibliotheken in Frage. Am besten dokumentiert ist jedoch mit Abstand *Pillow*. Diese Bibliothek ist außerdem perfekt für Python3 ausgelegt. Für die Darstellung der Bilder im Vollbildmodus eignet sich die Python Bibliothek *Tkinter*. Diese erweitert die Funktionen von *Pillow* um die Darstellungsebene auf dem Bildschirm des Users.

6.1.3 Berechnung von Breite und Höhe des finalen Bildes

Einen zentralen Punkt im Projekt stellt die Frage dar, wie die Größe des Tattoos von der Haut des Kunden abgelesen werden kann. Der Körper des Menschen weist an jeder Stelle Unebenheiten und Rundungen auf. Projiziert man also ein flaches, zweidimensionales Bild darauf, wird dieses zwangsläufig Verzerrungen aufweisen. Diese Tatsache stellt allerdings für die Idee des Projektes kein fatales Problem dar. Die Größe des geplanten Tattoos kann schon durch eine einzige Gerade, die die geplante Größe auf dem Körper hat, abgelesen werden.

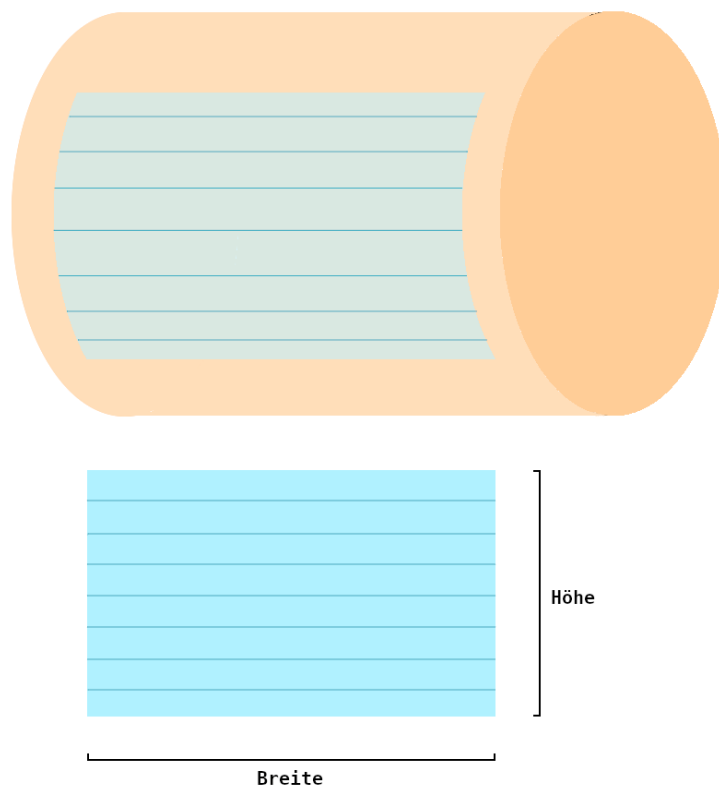


Abbildung 5: Lesbarkeit der Bildbreite

Leider wird das projizierte Bild durch die unebene Oberfläche verzerrt. Für eine weiterentwickelte Version des Prototyps könnte man mit zwei oder drei zusätzlichen Abstandssensoren eine Annäherung an die Wölbung der Haut des Kunden berechnen. Es könnte auch überlegt werden die Haut mit Lidar Sensorik abzutasten und die Haut und ihre Wölbungen so zu virtualisieren. Anschließend könnte man die Bilddatei als Textur auf das virtuelle Modell legen. Damit würde das Bild auch auf der gewölbten menschlichen Haut korrekt dargestellt werden. Für die Entwicklung des Prototyps und das Testen der praktischen Möglichkeiten des Gerätes im tatsächlichen Tagesgeschäft eines Tätowierers genügt aber das Messen eines einzelnen Tiefenpunktes. Die Einstellung der Größe funktioniert mit dieser Variante dennoch, da der User die Größe trotzdem anhand der einen Gerade abschätzen kann.

Der User hat generell zwei Möglichkeiten die Größe des projizierten Bildes zu beeinflussen. Das ist entweder die Verwendung des Scrollrades, also dem Drehgeber, oder die Veränderung des Abstands zur Haut des Kunden. In einem Bereich, dessen Grenzen nur durch die maximale Größe des Druckers begrenzt wird, kann mit diesen beiden Faktoren die Größe des geplanten Tattoos perfekt an die Wünsche des Kunden angepasst werden. Verlässt der User den messbaren Bereich, erhält er eine visuelle Rückmeldung vom Gerät.

Die Berechnung der neuen Größe des Bildes setzt sich aus dem Faktor d , also der Distanz zur Haut des Kunden und einem Faktor p , der Größenstufe am Scrollrad zusammen. Die Höhe des neu Berechneten Bildes H_{neu} , kann aus der neu berechneten Breite B_{neu} aus dem ursprünglichen Verhältnis V gewonnen werden. Das ursprüngliche Verhältnis setzt sich aus der alten Breite B_{alt} und Höhe H_{alt} zusammen.

$$V = \frac{H_{alt}}{B_{alt}}$$

$$B_{neu} = (B_{alt} + p) \cdot d$$

$$H_{neu} = B_{neu} \cdot V$$

Der Abstand des Projektors zur Haut muss mit dem alten Pixelsatz verrechnet werden. Proportional zum Abstand wächst auch die neue Größe des Bildes. Die zusätzlichen Schritte am Drehgeber werden als Pixelstufen auf die berechnete Größe addiert. Abschließend wird die Höhe des neuen Bildes aus dem Ursprungsverhältnis rückgerechnet.

6.1.4 Frequenz des Abstandssensors

Aus den oben genannten Formeln lässt sich schließen, dass für die Berechnung der neuen Größe eines Bildes nur ein einziges Mal ein Wert aus dem Abstandssensor gelesen werden muss. Allerdings sollte der User regelmäßig über die Einhaltung des messbaren Bereiches informiert werden. Deshalb sollte der Sensor in einem Kontrolltakt und zur Berechnung des finalen Bildes Werte messen. Die Frequenz der Messung sollte mindestens jede halbe Sekunde geschehen.

6.1.5 Auswahl des internen Datentyps

Das ursprüngliche Bild wird während der Prozedur im Prototyp mehrmals verändert. Seine Höhe und Breite wird vom User fortlaufend vergrößert und verkleinert. Damit das Bild während dieser Manipulationen keine Verluste erleidet, sollte ein Datentyp für die Zwischenspeicherung gewählt werden, der eine verlustfreie Kompression ermöglicht. Damit kann das JPEG Format für das Projekt ausgeschlossen werden. Da die Veränderung von PDF-Dateien nur über Umwege möglich ist, kann auch dieser Datentyp ausgeschlossen werden. Somit kommt also nur das Datenformat PNG für die internen Vorgänge im Prototyp in Frage.

6.2 Konzept

Die technische Spezifikation lieferte eine genaue Vorstellung über den fertigen Prototypen, die sich aus den Angaben der Kundin und den technischen Möglichkeiten ergibt. So lässt sich ein Plan für das technische Konzept der Hardware erstellen. Für die Steuerung des Prototyps soll ein Raspberry Pi 3 verantwortlich sein. Voraussichtlich liegt der Rechenschwerpunkt der Software in der Messung des Abstands und der Berechnung der unterschiedlichen Bildgrößen. Das sollte der Raspberry Pi problemlos bewältigen können.

Für die Auswahl der Projektorplatine ist vor allem die Helligkeit wichtig. Der Beamer muss dazu in der Lage sein, über eine Distanz von ungefähr 1,5 Metern ein Bild auf den Körper eines Kunden zu projizieren. Dies sollte auch bei Tageslicht oder in einem künstlich beleuchteten Raum überzeugend funktionieren. Die ausgewählte Platine von Texas Instruments erzeugt mit seiner nHD-Optik-Engine bis zu 30 Lumen und sollte also für diese Zwecke ausreichen. Die Platine ist eigentlich für den Raspberry Pi Konkurrenten BeagleBone ausgelegt. Die Kleinstcomputer von BeagleBone sind allerdings nicht WLAN fähig und kommen deshalb für den Prototypen nicht in Frage. Um die Projektorplatine mit dem Raspberry zur verbinden werden im späteren Verlauf voraussichtlich einige zusätzliche Kabel und Verlotungen notwendig.

Der Abstandssensor sollte, wie die Projektorplatine, einen Bereich von bis zu 1,5 m unterstützen. Ein Time-of-Flight Sensor hat das Problem, dass seine Messungen mit größeren Distanzen ungenauer werden. Dennoch ist diese Art von Sensor die kostengünstigste und naheliegendste Option. Es ist nicht ausgeschlossen, dass im späteren Verlauf klar wird, dass der ausgewählte Sensor mit der Anforderung überfordert ist. Um die Ausrichtung des Abstandssensors zu überprüfen, wird zusätzlich ein Laserpointer an der Vorderseite des Prototyps angebracht. Dieser zeigt genau in welche Richtung der Abstandssensor misst.

6.2.1 Hardware

Bauteil	Aufgabe
DLP LightCrafter Display 2000 EVM	Projizieren der Bilder
Raspberry Pi 3	Steuerung der Oberfläche und Sensorik
VL53LOXV2	Messen des Abstands zur Haut
Grove Mouse Encoder	Einstellung der Bildgröße

Tabelle 1: Die Bauteile und ihre Aufgabenbereiche

6.2.2 Software

Der Aufbau der Softwarestruktur kann ebenfalls aus der funktionellen Spezifikation gewonnen werden. Für die Architektur der Software bietet sich eine Datenpipeline mit drei Schichten an.

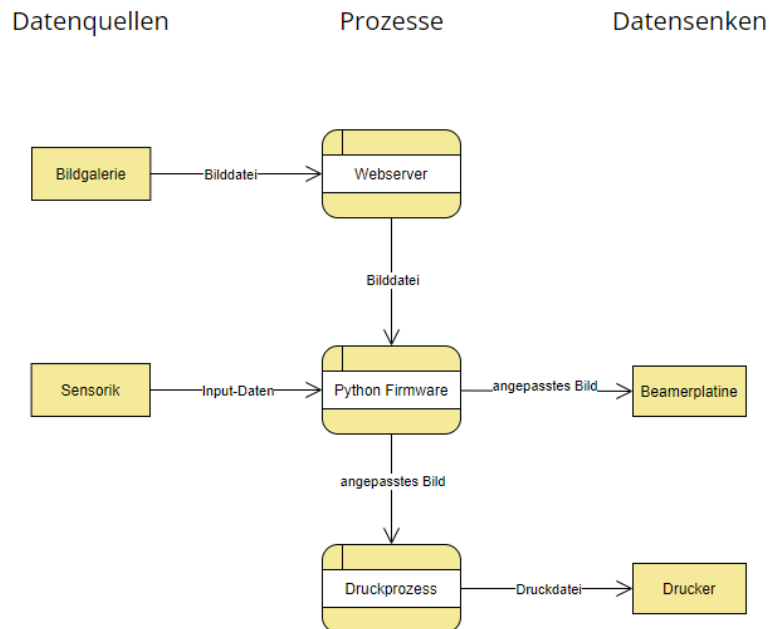


Abbildung 6: Software-Schichten des Prototyps

Die erste Schicht ist die Webserver-Schicht. Diese ist für die Oberfläche des Prototyps verantwortlich. Sie erhält über die zur Verfügung gestellte Website aus der Bildgalerie des Users eine Bilddatei. Die Bildgalerie des Users ist eine der zwei Datenquellen in diesem System. Die Webserver Schicht speichert die Datei lokal und stellt diese so für die zweite Schicht des Systems zur Verfügung. Diese ist die Firmware-Schicht. Sie erhält neben der Bilddatei außerdem Daten aus dem Time-of-Flight Sensor. Die Sensorik stellt die zweite Datenquelle des Systems dar. Die Firmware Schicht verarbeitet die erhaltenen Daten und gibt diese an zwei Empfänger weiter. Die Eine ist die Projektorplatine, eine der zwei Datensenken des Systems. Außerdem schickt die Firmware das Bild an die letzte Schicht, den Druckprozess. Diese Schicht ist für das Drucken mit dem vom User ausgewählten Stencil-Drucker verantwortlich. Die Druckschicht verbindet sich mit dem Drucker und erstellt dort einen Druckauftrag mit dem angepassten Bild. Der Drucker ist hierbei die letzte Datensenke des Systems.

Programmiersprache	Schicht	Aufgabenbereich
Python	Firmware, Druckprozess	Anzeige, Berechnungen
C	Firmware	Sensorik
Html & Css	Webserver	Oberfläche
JavaScript	Webserver	Datentransfer
PHP	Webserver	Datentransfer

Tabelle 2: Die Programmiersprachen und ihre Aufgabenbereiche

6.2.3 Gehäuse

Für das Gehäuse des Prototyps wird mit einem Grafikprogramm eine Skizze erstellt. Diese wird nach den, in der funktionellen Spezifikation festgehaltenen Wünschen der Tätowiererin vorbereitet. Wichtig ist es hierbei, dem Prototypen ein modernes Aussehen zu verleihen, damit dieser als technische Neuerung erkannt wird.

Sobald der Prototyp von technischer Seite aus funktioniert und getestet ist, wird das Gehäuse mit dem kostenlosen 3D Modelling Programm *FreeCAD* konstruiert. Die Kühlung der Projektorplatine muss bei der Modellierung des Gehäuses unbedingt berücksichtigt werden, genauso wie die Lage des Abstandssensors und der anderen Sensorik. Das so entstandene 3D-Modell wird dann an einem 3D Drucker ausgedruckt.

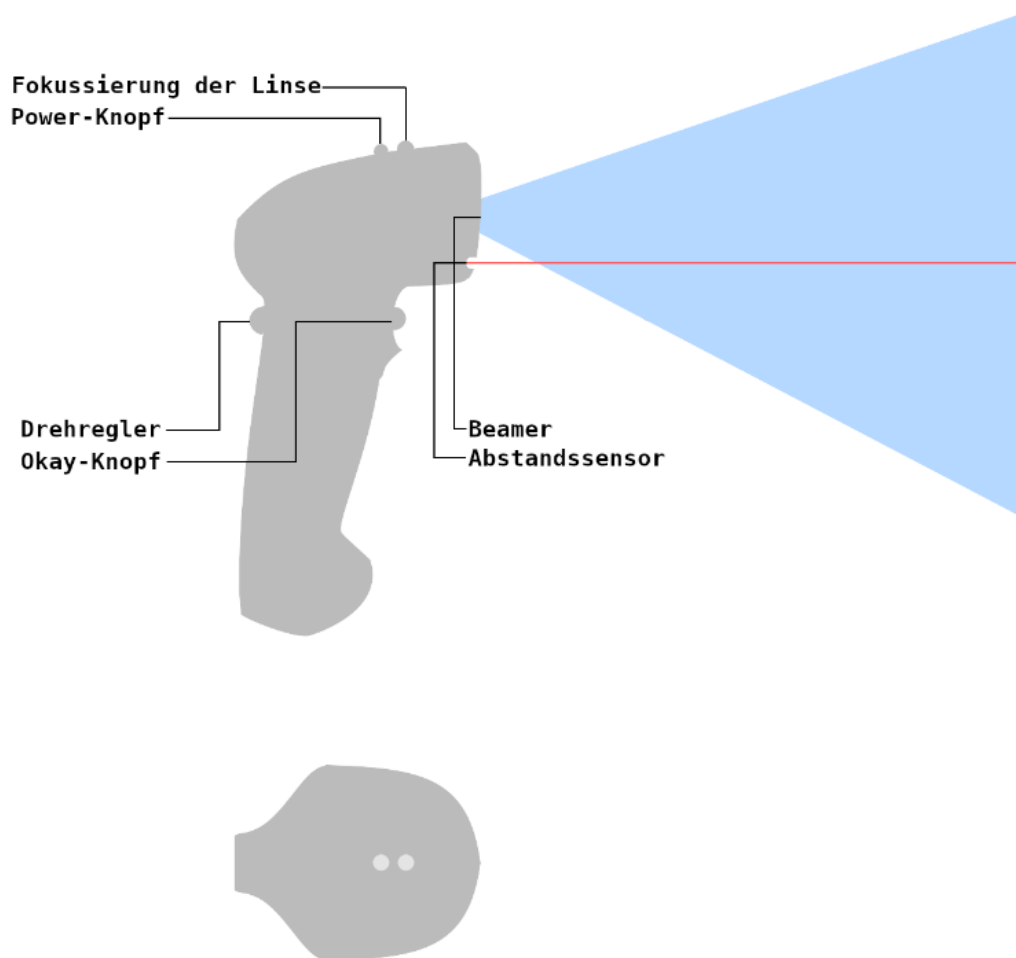


Abbildung 7: Visuelle Erscheinung mit Seiten- und Draufsicht

7 Dokumentation der Implementierung

7.1 Emulieren eines Druckers

Der erste Schritt in der Umsetzung des Projektes ist das Maskieren des Raspberry Pi als ein Netzwerkdrucker. Damit sollten alle Geräte im Netzwerk den Prototypen finden und ihm Druckaufträge zuschicken können. Aus dem Druckauftrag soll dann das zu verarbeitende Bild entschlüsselt und lokal gespeichert werden.

Die Recherche zum Thema Druckprotokoll zeigte, dass Netzwerkdrucker mit dem Internet-Printing-Protocol kommunizieren. Dieses ermöglicht die Bearbeitung und das Erstellen von Druckjobs über das Internet. Hat man also vor einen Netzwerkdrucker mit dem Raspberry Pi zu emulieren, so muss man die Druckaufträge aus dem Netzwerk mithilfe der CUPS Methoden empfangen und entschlüsseln.

7.1.1 Evaluation

Die Recherche ergab außerdem, dass die Bibliotheken, die in C oder Python eingebunden werden können, ausschließlich Druckaufträge senden können. Das Empfangen von Druckaufträgen ist in keiner der Libraries vorgesehen. Das bedeutet also, um einen Drucker zu emulieren müsste man mühsam die gesendeten und empfangenen Pakete überwachen, um diese anschließend von Hand nachzubauen. Hier erschwert einem die gut geschützte Software der Apple-Geräte zusätzlich die Arbeit. Außerdem werden die Druckdaten als PPD-Files verschickt, einem von Adobe entwickelten Datentyp, speziell für Druckaufträge. Die Bilder werden so zusammen mit Formatinformationen über das Netzwerk verschickt. Um diese Dateien zu entschlüsseln wäre es nötig, eine sehr komplexe Entschlüsselungs-Software zu entwerfen. Diese sollte zusätzlich mit verschiedenen Geräten kompatibel sein. Die Idee, mit dem Raspberry einen Drucker zu emulieren stellt sich als deutlich komplexer heraus, als erwartet.

7.1.2 Fazit

Für die Programmierung des Prototyps wird auf das Erstellen einer IPP-Schnittstelle zwischen dem Prototyp und dem Tablet des Users verzichtet. Die Dateien gelangen also nicht wie geplant über das Senden eines Druckauftrages zum Prototyp, sondern müssen vom User auf der Weboberfläche hochgeladen werden. Möglicherweise wird auf die Idee der Druckeremulation bei der Erweiterung des Prototyps zu einem marktfähigen Produkt zurückgegriffen.

7.2 Verdrahtung der Bauteile

Bevor die Programmierung des Prototyps beginnen kann, müssen die Bauteile korrekt miteinander verdrahtet werden. Außerdem müssen die Konfigurationsdateien des Raspberry Pi auf den Betrieb der Projektorplatine eingestellt werden. Die GPIO Pins des Computers müssen entsprechend belegt und die Konfiguration für das I2C-Protokoll muss entsprechend angepasst werden. Der Raspberry Pi bildet den Grundbaustein des Projektes. An seinen Datenpins werden alle anderen Bauteile zusammengeführt.

7.2.1 Abstandssensor

Nachdem die vier Pins für den kleinen Time-of-Flight Sensor erfolgreich an die Kabel gelötet sind, müssen diese mit den entsprechenden GPIO Pins am Raspberry Pi verbunden werden. Für VCC und GND hat der Raspberry ausreichend Plätze. Die zwei Datenpins jedoch, SCL und SDA, gibt es am Raspberry nur jeweils einmal. Da sowohl Projektorplatine als auch der Abstandssensor diese Pins verwenden muss hier eine Lösung gefunden werden. Da das I2C Protokoll für mehrere Client Geräte funktioniert, ist die naheliegendste Lösung das Löten eines Busses für die beiden Pins.

Sowohl SCL als auch SDA spalten sich nach dem Löten in zwei verschiedene Kabel auf. Eins der Enden ist für den Abstandssensor, das andere Ende ist für die Kommunikation mit der Projektorplatine vorgesehen. Somit können beide Clients über das I2C-Protokoll angesprochen werden.

Ein kurzer Test des Sensors zeigt, dass der Sensor Daten sendet. Ändert man den Abstand zum Sensor, verändern sich auch die Daten proportional. Ob der Wert tatsächlich mit der realen Größe übereinstimmt, muss in einem späteren Kalibrierungsschritt getestet und korrigiert werden.

7.2.2 Projektorplatine

Für die Projektorplatine findet sich online eine Mapping der Raspberry Pins auf die Projektor-Pins. Die Verkabelung an sich stellt kein großes Problem dar. Die Projektorplatine bezieht seine benötigten 5V entweder per Jumperkabel oder direkt mit einem Netzteil aus der Steckdose. Doch obwohl die Verkabelung mehrfach überprüft und kontrolliert wurde, verhält sich die Platine nicht wie erwartet. Sie projiziert immer und immer wieder nur den Start Screen des Herstellers und nicht den Bildschirm des Raspberry Pi. Als Ursache des Problems kommen einige Fehlerfaktoren in Frage.

Zum einen zeigt der Raspberry nach dem Boot dauerhaft eine Warnung für Niedrigspannung an. Das kann dazu führen, dass die GPIO Pins sich nicht wie erwartet verhalten und die Sensorik in niedriger Frequenz taktet. Deshalb wurde für die Stromversorgung des Raspberry ein stärkeres Netzteil besorgt. Dieses hat nun 3 Ampere und ist damit deutlich stärker als die meisten USB-Netzteile. Die Warnung wurde daraufhin zwar seltener angezeigt, aber behoben schien das Problem noch nicht. Auch die Projektorplatine hing weiterhin in ihrem Startbildschirm fest.

Ein zweiter Fehlerfaktor kann die Konfiguration des Raspberry sein. Falls bestimmte essenzielle Konfigurationseinstellungen von anderen Dateien oder Programmen überschrieben werden, kann das zum Fehlverhalten der Platine führen. Doch auch langes Korrigieren und Ändern verschiedener Pin-Belegungen erzielte hier kein Ergebnis.

Der letzte Fehlerfaktor entstammt dem Kommunikationsprotokoll, mit dem sowohl der Abstandssensor als auch der Projektor kommunizieren. Das I2C-Protokoll ist leider recht anfällig für Konfigurationsfehler und sehr schwierig zu durchblicken. Der Konsolenbefehl `i2cdetect` liefert Auskunft über die vom Master gefundenen Slave-Geräte. Der Raspberry scheint die Platine nicht als Kommunikationsakteur wahrzunehmen. Der Abstandssensor wurde als I2C-Client erkannt und wird auf dem Hexadezimalen Platz 0x29 angezeigt. Das ist die eindeutige Adresse des Abstandssensors.

```
beamer@raspberrypi:~ $ i2cdetect -l
i2c-11  i2c          ffffffff.i2c          I2C adapter
i2c-2   i2c          bcm2835 (i2c@7e805000) I2C adapter
beamer@raspberrypi:~ $ i2cdetect -y 11
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --
beamer@raspberrypi:~ $
```

Abbildung 8: Darstellung des I2C-Ports 0x29

Die Platine jedoch scheint weiterhin Probleme zu bereiten. Für die Verkabelung des Abstandssensors wurde, wie erwähnt, ein Bus gelötet, um mehrere Entitäten an den zwei I2C Pins des Raspberry anzuschließen. Der Bus scheint jedoch nur einen der Akteure zu erkennen. Aus einer Recherche zum Thema geht hervor, dass die einfachste Lösung für das Problem wahrscheinlich das Erstellen eines neuen i2c Busses an zwei anderen GPIO-Pins des Raspberry darstellt. Damit wären die Kommunikationsprobleme hinfällig. Auf diese Weise kann es in keinem Fall zu Verständigungsproblemen zwischen Sensor, Projektor und Raspberry kommen, da verschiedene Leitungen verwendet werden

Nachdem zwei neue Pins für die Kommunikation mit dem Abstandssensor verantwortlich gemacht wurden und eine weitere Änderung in der Konfiguration des Raspberry vorgenommen wurde, zeigt der Projektor endlich, wie gewünscht den Desktop des Raspberry Pi. Auch der Abstandssensor sendet wieder erfolgreich Daten. Damit sind die Weichen gestellt für den nächsten Schritt.

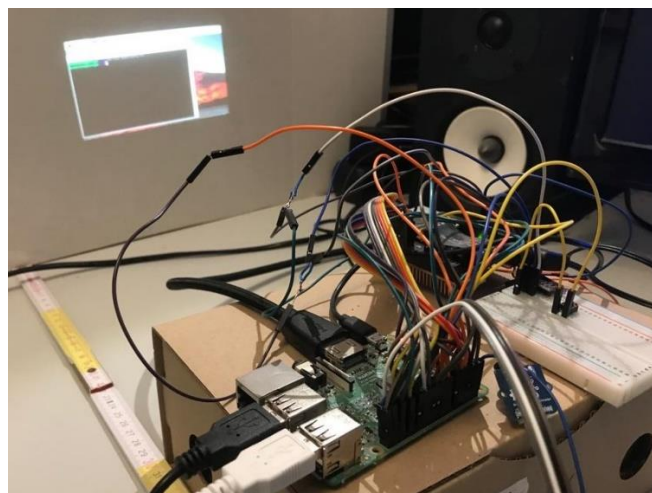


Abbildung 9: Elektronik der Projektorplatine

7.2.3 Drehimpulsgeber

Der Drehimpulsgeber wurde vom Hersteller leider ohne die nötige Achse geliefert. Deshalb muss die Achse per 3D Druck nachgebaut werden. Nach der Vermessung der Achse des Drehimpulsgebers und dem Modellieren des benötigten Bauteils kann die Achse eingesetzt werden. Für die Verkabelung des Sensors sind 4 Pins verantwortlich. GND und VCC, sowie zwei digitale Pins für Signal A und B. Die Verkabelung des Drehimpulsgebers ist unproblematisch, da keine zusätzlichen Kommunikationsprotokolle zur Messung benötigt werden. Das Auslesen der Drehgeberdaten mit den GPIO-Pins des Raspberry funktioniert glücklicherweise auf Anhieb problemlos.

7.2.4 Fazit

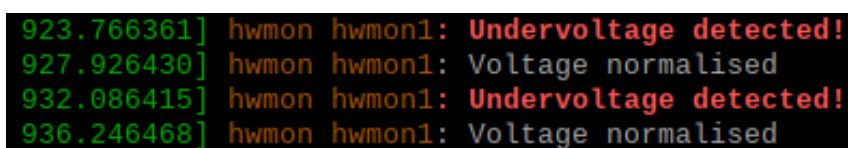
Das Verdrahten der Bauteile nahm sehr viel Zeit in Anspruch. Das Verkabeln der Projektorplatine wurde durch die schlechte Dokumentation des Bauteils zusätzlich erschwert.

7.2.4.a I2C-Problembehandlung

Zwar ist das I2C-Protokoll durchaus in der Lage, mehrere Ein- und Ausgabe- Geräte an einem Bus zu bedienen, allerdings hat sich das Debugging der Variante mit einem Bus als zu zeitaufwändig herausgestellt. Die Lösung des Problems schien das Erstellen eines zweiten Busses für den Abstandssensor zu sein. Glücklicherweise stellt der Raspberry gradeso genügend Pins für alle Geräte zur Verfügung, so dass die Idee umgesetzt werden konnte. Die Pins, die für die I2C-Kommunikation benutzt werden sollen, müssen sehr schnell zwischen Hoch- und Niedrigspannung hin und her springen. Das Protokoll sendet ständig Daten zwischen Client und Sensor. Es stellt sich heraus, dass die GPIO-Pins eine geringere Datenfrequenz aufweisen als die I2C-Pins des Raspberry. Das führt dazu, dass das projizierte Bild stark ruckelt und unlesbar ist. Deshalb wird die Idee eines gemeinsamen Busses für beide Clients erneut aufgegriffen und umgesetzt. Beim Debuggen dieses Ansatzes und dem Schrauben an den I2C Geräten muss die Funktionstüchtigkeit des Projektors ständig getestet und überprüft werden, damit der Videodatenfluss weiterhin funktioniert. Glücklicherweise zeigte sich, dass der Fehler in der I2C-Konfiguration lag. Beide I2C-Clients kommunizieren jetzt fehlerfrei mit dem I2C-Master, dem Raspberry.

7.2.4.b Niedrigspannung

Der Raspberry Pi zeigt weiterhin gelegentlich die Warnung, dass die angelegte Spannung zu niedrig sei. Jedoch konnte während der Entwicklung keine Beeinträchtigung durch diese Tatsache festgestellt werden. Um diese Warnung zu beseitigen, wäre wahrscheinlich ein noch stärkeres Netzteil von Nöten. Die Hardware kann auf Dauer Schaden durch die Niedrigspannung nehmen. Deshalb muss diese Warnung bei einem fertigen kommerziellen Produkt behoben werden. Für den Prototypen ist das jedoch weniger relevant und die Fehlermeldung kann ignoriert werden.



```
923.766361] hwmon hwmon1: Undervoltage detected!  
927.926430] hwmon hwmon1: Voltage normalised  
932.086415] hwmon hwmon1: Undervoltage detected!  
936.246468] hwmon hwmon1: Voltage normalised
```

Abbildung 10: Angezeigte Fehlermeldung zur Niedrigspannung

7.2.4.c Helligkeit

Der Projektor strahlt so hell, dass selbst bunte Bilder in künstlich beleuchteten Räumen sehr klar und erkennbar sind. Damit ist ein Austauschen dieses Bauteils nicht von Nöten und die Implementierung der Software kann beginnen.

7.3 Oberfläche

Das Erstellen einer geeigneten Oberfläche für den Prototypen ist ein zentraler Teil der Nutzbarkeit des Gerätes. Aus diesem Grund ist es wichtig, diese schlicht und modern zu gestalten. Es darf keine langen Ladezeiten geben und die Dateiverwaltung muss einwandfrei und ohne Probleme funktionieren. Um die Bilder des Users zu speichern, benötigt der Server einen Ordner */uploads* der die hochgeladenen Dateien speichert. Aus diesem Ordner bedient sich die Firmware des Prototyps bei der weiteren Verarbeitung der Bilddateien. Nach der Benutzung des Gerätes muss sich der Ordner außerdem vollständig leeren, damit der interne Speicher des Raspberry Pi nicht unnötig belastet und die Ordnung bewahrt wird. Der Projektor arbeitet immer mit genau einem Bild. Dieses wird erst gespeichert und anschließend verarbeitet. Dieses System ist auch gut erweiterbar. Im späteren Verlauf kann überlegt werden, eine kleine User-Bildgalerie zu erstellen, die beispielsweise bis zu 10 Bilder speichern kann. So kann der Vorgang des Hochladens der Bilder gegebenenfalls beschleunigt werden.

7.3.1 Website

Die Website setzt sich aus einer HTML-Landingpage, einer CSS-Datei und einem JavaScript zusammen. Diese Dateien werden beim Besuchen der Landingpage dann vom Browser des Users heruntergeladen.

7.3.1.a HTML und CSS

Der erste Entwurf der Website soll vor allem zum Testen der Ajax Bildübertragung und der Python Skripte auf dem Server dienen. Daher genügt eine simple Oberfläche mit einem Upload Button. Über diesen soll der User sein Bild auswählen und es an den Server schicken können. Spätere Versionen der Website sollen einen moderneren Look bekommen sowie ein Drop-Down-Menü, über das der User den richtigen Drucker auswählen kann. Außerdem soll die Website ein Ausgabefeld für die gemessene Distanz zur Haut des Kunden bekommen, damit der User den richtigen Abstand zur Haut beibehalten kann. Um die Nutzbarkeit der Oberfläche optimal zu gewährleisten, bleibt die Website recht schlicht und übersichtlich. Die erste Variante der Website wurde mit der Tätowiererin besprochen und auf ihre Wünsche hin wurden kleine Änderungen vorgenommen.

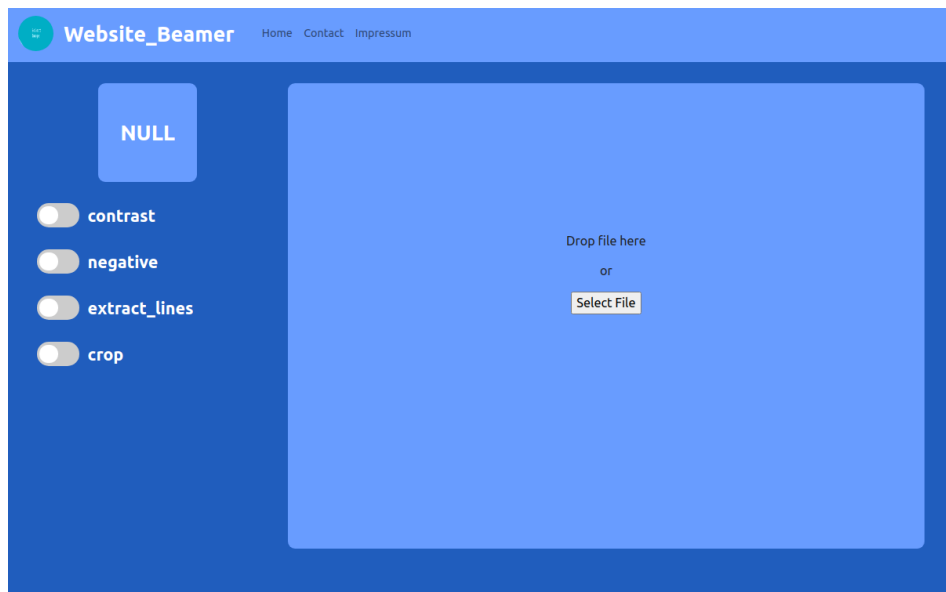


Abbildung 11: Web-Oberfläche des Prototyps

Die Oberfläche des Prototyps wurde zudem um eine Schaltfläche im linken Bildschirmbereich erweitert. Über diese kann der User gewünschte Manipulationsschritte für sein hochgeladenes Bild anwählen. Damit verfügt er über die Möglichkeit den Kontrast zu erhöhen, das Bild zu invertieren und automatisch leere Teile des Bildes wegzuschneiden. Dieses Feature erhöht zusätzlich die Performance der späteren Prozesse, da in diesen dann nicht mit leeren weißen Pixeln gerechnet werden muss. Der letzte Regler ermöglicht es, das hochgeladene Bild durch einen selbstprogrammierten Filter so zu manipulieren, dass die dunkelsten Bereiche isoliert werden und alle hellen Bereiche mit weiß gefüllt werden. Dieser „Linienextraktor“ soll bei der Nutzung des Prototyps mit Stencil Druckern helfen, da diese oft große Probleme mit Grauwerten haben.

Zur Vereinfachung der strukturellen Einteilung der Seite ist Bootstrap3 in die HTML Datei eingebunden. Diese Bibliothek ermöglicht es, ein modernes Design für kleinere Displays wie Handys oder IPads zu gestalten. Nach fertigem Upload des Bildes ruft das Frontend die hochgeladene Ressource vom Server ab und kontrolliert somit dessen Richtigkeit. Das so angefragte Bild wird dann auf der Website dargestellt, um dem Kunden das erfolgreiche Hochladen zu bestätigen.

7.3.1.b JavaScript

Der erste Teil des JavaScripts ist für den Upload der Dateien verantwortlich. Hier wird außerdem der Explorer des Users an die Website angeknüpft, damit die Bilddatei bequem ausgewählt werden kann. Das JavaScript schickt außerdem GET und POST Anfragen an den Webserver, mit denen Bilder hochgeladen und gelöscht werden können.

Hat der User sein Bild ausgewählt, schickt das JavaScript einen ersten GET- Request an den Server. Das angefragte PHP-File löscht dann eventuelle alte Bilder vom Server. Anschließend übermittelt das JavaScript die ausgewählten Flags für die Bildmanipulation. Auch diese Daten werden mit einem einfachen GET Request ans Backend übermittelt. Das angesprochene PHP File schreibt die ausgewählten Einstellungen dann in eine auf dem Server gespeicherte Textdatei. Diese Textdatei ist die zentrale Konfiguration des Prototyps.

Der eigentliche Upload der Bilder erfolgt mit Hilfe des Ajax-Protokolls. Dieses ermöglicht den Datenaustausch vom Gerät des Users zum Server. Signalisiert die Website, dass der User die Daten vollständig ausgefüllt hat, werden die Daten in ein Array geschrieben. Dieses wird dann Stück für Stück an den Server gesendet. Dieser dekodiert das empfangene Array dann in die benötigten Daten. Während der Implementierung stellte sich heraus, dass Bilder ab einer gewissen Größe nicht gesendet werden konnten. Nach einer Recherche zeigte sich, dass die PHP-Konfiguration dafür verantwortlich ist. Diese enthält für die Variable *upload_size* standardmäßig den Wert *2M*, also 2 Megabyte maximaler Upload-Größe. Glücklicherweise kann diese Voreinstellung durch Veränderung der PHP-Konfiguration überschrieben werden. In diesem File werden die Voreinstellungen festgehalten, die bei Betrieb des Servers überschrieben werden sollen.

7.3.2 Webserver

Um den Raspberry Pi als Netzwerkakteur erscheinen zu lassen, fiel die Entscheidung auf einen Apache2 Webserver, da dieser auf Linux Systemen sehr einfach nutzbar ist. Die Methoden des Webserverns verteilen sich auf mehrere Skripte, die jeweils Teile der Funktionen bewältigen.

Insgesamt drei PHP-Skripte beantworten die Anfragen, die das Frontend an den Server schickt. Das erste PHP-Skript empfängt den Namen eines Bildes aus der URL und löscht dieses aus dem Upload-Ordner. Das Skript kann zudem, falls gewünscht, den kompletten Server von unnötigen Dateien bereinigen. Das zweite PHP-Skript empfängt die Einstellungen zur Bildmanipulation und schreibt diese in ein Textdokument. Um die Sicherheit des Systems zu gewährleisten, werden gelesene Strings auf ihre Gültigkeit überprüft. So können Codeinjektionen verhindert werden.

Das letzte PHP-Skript wird vom JavaScript immer dann aufgerufen, wenn der Server ein Bild empfangen soll. Im PHP wird dann als erstes der Dateityp der zu empfangenden Datei überprüft. Hier werden ausschließlich verschiedene Formen von PNG's und JPEG's akzeptiert. Es gibt heute viele Varianten dieser Datentypen, die in unterschiedlichen Bereichen Anwendung finden (Miano, 1999). Anschließend wird die Datei aus dem Ajax Array ausgelesen und in den entsprechenden Dateityp abgelegt. Der Name der Datei kann dabei aus der übertragenen Datei gelesen werden. Die abgelegte Datei wird auf der Festplatte des Servers gespeichert und kann dort von der restlichen Software gelesen werden.

7.3.3 Fazit

Das Erstellen der Web-Schnittstelle stellte glücklicherweise kein großes Problem dar. Hier konnten alle benötigten Funktionen schnell und simpel implementiert werden. Die Schaltfläche zur Bildmanipulation wurde erst während der Programmierung der Firmware-Schicht hinzugefügt, da diese vorher nicht von Nöten war. Das Lesen der PHP-Dokumentation half dabei die Hürden der Backend-Programmierung schnell zu überbrücken.

7.4 Firmware

Als Firmware des Prototyps wird im Folgenden die Software genannt, die für das Anzeigen der Bilder und das Verrechnen der anderen Daten der Sensorik zuständig ist. Diese Software ist in Python geschrieben und erfüllt alle rechnerischen Aufgaben des

Gerätes dauerhaft, während dieses angeschaltet ist. Der Fokus bei der Programmierung der Firmware liegt auf der möglichst performanten Umsetzung der Funktionen. Viele Teile der Programme wurden deshalb mehrmals neu geschrieben und im Laufe der Zeit verbessert. Die Firmware des Prototyps wird mit dem Boot des Betriebssystems des Raspberry gestartet und läuft bis zum Ausschalten durchgängig.

7.4.1 Darstellung der Dateien

Die Darstellung der Dateien wird durch die im Rechercheteil gefundene Python Bibliothek *Tkinter* ermöglicht. Diese Bibliothek kann in einem eigenen GUI-System die Bilder im Fullscreen Modus auf dem Display des Raspberry anzeigen. Da der Projektor den Desktop des Raspberry spiegelt, wird dadurch das jeweilige Bild vom Projektor im Vollbildmodus ausgestrahlt. Wird der Drehgeber während der Darstellung gedreht, berechnet das Python Programm die neue Größe des Bildes in Abhängigkeit von Breite und Höhe des Ursprungsbildes. Das so berechnete Bild wird anschließend an die Projektorplatine geschickt und ausgestrahlt. Die Veränderung der Bildgröße wird durch die Bibliothek *Pillow* für Python3 ermöglicht. Nach ersten Tests stellt sich jedoch heraus, dass das Verändern der Bildgröße für den Raspberry Pi einen starken Rechenaufwand mit sich bringt. Obwohl die Belastung des physischen Speichers durch weitere Verminderungen der Schreib und Lese Zugriffe vermindert wurde, stockt die Berechnung der neuen Bilder bei größeren Breitewerten weiterhin massiv.

Es liegt nahe, dass die Performance-Schwäche an der Methode von *Pillow* liegt oder der Raspberry einfach an die Grenzen seiner Rechenleistung stößt. Eine Recherche zum Thema der performanten Bildgrößenänderung konnte noch einige mögliche Verbesserungen ans Licht bringen.

Ein Entwickler verfasste 2017 einen Artikel über seine eigens entwickelte Bildmanipulations-Software. Die Python-Bibliothek mit dem Namen *Pillow-SIMD* wird von seinem Entwickler Alex Karpinsky als 15-mal schneller als *Pillow* beworben (Karpinsky, 2017).

Zusätzliche Performance-Verbesserungen können durch das Verwenden der *Numpy*-Bibliothek erzielt werden. Die in der Library enthaltenen Arrays schreiben Daten etwas schneller als die Python üblichen Listen.

Glücklicherweise kann der User die Größe des Bildes auch durch Verändern des Abstandes zum Kunden verändern. Der Drehgeber ermöglicht ihm also nur eine weitere Möglichkeit zur Größenmanipulation. Aus diesem Grund ist die Verbesserung der Bildvergrößerungsfunktion vorerst wenig relevant.

7.4.2 Optionen zur Bildoptimierung

Um die Sichtbarkeit der projizierten Bilddateien zu verbessern, wurden zusätzlich einige vom User auswählbare Manipulationsschritte in die Weboberfläche eingebaut. So lässt sich durch das Betätigen des entsprechenden Reglers auf der Web-Oberfläche beispielsweise der Kontrast des Bildes verbessern. Das Bild kann außerdem invertiert werden und wird dann grafisch negativ dargestellt. Die Software überprüft vor der Darstellung des Bildes jedes Mal, welche vom User ausgewählten Manipulationsschritte vorgenommen werden sollen und hält den gewünschten Zustand lokal fest. Die Methoden für diese Manipulationsschritte entstammen, bis auf den selbstgeschriebenen Linienextraktor, der *Pillow* Bibliothek.

7.4.3 Empfangen und Glätten der Abstands Daten

Die Daten des Sensors im Parallelbetrieb mit dem Projektor auszulesen, stellt sich als deutlich komplexer heraus als während der Verkabelung angenommen. Zwar werden beide I2C-Geräte vom Betriebssystem erkannt, aber die Daten des Abstandssensors kollidieren mit dem ständigen Datenfluss an den Projektor. Die Daten, die der VI53I0x mit der Time-of-Flight-Methode liest und auf den I2C-Port schickt sind also nicht lesbar. Das Problem lässt sich mit einem Subprozess in Python umgehen, welcher dann ein C-Programm, welches vom Hersteller zur Verfügung gestellt wird, aufruft. Die Daten können dann vom Python Haupt-Thread aufgefangen werden und gelangen so in den virtuellen Speicher des Python Prozesses. Um die Daten lesen zu können, war einige Arbeit und viel Ausprobieren nötig. Alle Versuche, die Daten mit Python direkt auszulesen, scheiterten an der I2C Schnittstelle.

Damit der Abstandswert möglichst nah an den realen Wert herankommt, wird nicht nur einmal der Abstand gelesen, sondern insgesamt 50 Mal. Eine kleine Testreihe zeigt, dass die Daten des Sensors sehr stark schwanken. Die Abstandswerte müssen also durch einen Filter geglättet werden. Der Filter sollte einige Variablen zur manuellen Kalibrierung zur Verfügung stellen und er sollte die Daten auf eine für den Prototypen optimale Weise manipulieren. Die Variablen des Filters sind *max_abweichung*, *trim*, *mindestabstand* und *maximalabstand*. Dazu wird als erstes die Standardabweichung der Werte errechnet. Übersteigt der Wert der Standardabweichung einen vorher festgelegten Wert *max_abweichung*, erhält der User eine Rückmeldung vom Prototyp. Der User wird gebeten das Gerät ruhig zu halten, damit die Daten gemessen werden können. Anschließend werden die in einem Array gesicherten Abstanddaten sortiert. Jetzt werden von beiden Enden des Arrays so viele Werte abgeschnitten, wie die Variable *trim* vorgibt. Damit können die größten Ausreißer in der Datenreihe eliminiert werden. Aus den restlichen Daten wird nun der Durchschnitt gebildet. Der daraus gewonnene Wert ist der momentane Abstandswert. Abschließend bekommt der User noch Rückmeldung, ob dieser Wert innerhalb des vorher festgelegten Bereiches zwischen *mindestabstand* und *maximalabstand* liegt. Die Rückmeldung für den User soll sowohl über die Website als auch die GUI des Prototyps visualisiert werden.

7.4.4 Verrechnen der Bilder mit den Sensor Daten

Der *Linsenfaktor f* des Projektors geht aus der verhältnismäßigen Vergrößerung des projizierten Bildes bei regelmäßigen Distanzsprüngen hervor. In einer Testreihe wurden jeweils die Distanz und die Breite des projizierten Bildes gemessen.

Distanz (cm)	Breite des projizierten Bildes (cm)
30	14,7
35	16,8
40	19
45	21,3
50	23,6
55	25,7
60	27,9

Tabelle 3: Manuelle Messung von Distanz und Breite des Bildes

Aus diesen Werten errechnet sich ein mittlerer Differenzwert von 2,2 cm pro 5 cm Distanz. Auf den Zentimeter gerechnet ergibt sich damit eine mittlere Differenz f von 0,44 cm. Dieser Wert spiegelt den Linsenfaktor und wird für die finale Bildberechnung benötigt. Da der Wert aus vom Menschen gemessenen Daten hervorgeht, muss mit einem gewissen Fehler gerechnet werden. Im Kalibrierungsschritt wird dieser Wert überprüft und gegebenenfalls minimal angeglichen werden, damit die Distanzumrechnung fehlerfrei funktionieren kann.

Da der Drucker Bilder im A4 Format druckt, ist die finale Breite und die finale maximale Pixelbreite des Bildes gegeben. Das in Deutschland gängige A4 Format ist an seiner längeren Seite 29,7 cm lang und besitzt 3508 Pixel. Das Papier wird hierbei horizontal gemessen, da auch das vom Projektor ausgestrahlte Bild ein 16:9 Format hat.

Aus diesen Formatvorgaben errechnet sich eine Pixeldichte von 118,8888 Pixeln pro cm, was 300 DPI entspricht.

Da die Größe des final berechneten Bildes die eines A4 Blattes nicht überschreiten darf, lässt sich mithilfe des vorher berechneten Faktors die maximale Distanz des Prototyps berechnen. Bei dieser Distanz sollte ein gestrahltes Bild genau in der Größe eines A4-Blattes gedruckt werden. Der Faktor x beschreibt in der folgenden Formel die Anzahl von cm, die zum Erreichen eines vollen A4 Blattes nötig sind. Die maximale Distanz d_{max} lässt sich dann aus einem beliebigen Startwert und dem Faktor x errechnen. Verwendet man einen beliebigen Startwert d_{sensor} , kann mithilfe der Breite eines A4 Blattes B_{a4} die Formel nach x umgestellt werden.

$$\begin{aligned}
 B_{a4} &= d_{sensor} + f \cdot x \\
 29,7cm &= 14,7cm + 0,44 \cdot x \\
 x &= 34,0909cm \\
 d_{max} &= x + B_{sensor} \\
 d_{max} &= 34,0909cm + 30cm \\
 d_{max} &= 64,0909cm
 \end{aligned}$$

Mit diesen Formeln ergibt eine maximale Distanz von 64,09 cm. Die Kalibrierung des Linsenfaktors f wirkt sich im späteren Verlauf auch auf die Maximaldistanz aus und andersherum. Übersteigt der User beim Verwenden des Prototyps diese Distanz, erhält er die Rückmeldung die Distanz zum Objekt zu verringern.

Um das finale Bild zu berechnen, muss von diesem maximalen Distanz-Wert rückwärts gerechnet werden. Der lineare Faktor f wird auf die Differenz zum Maximalwert multipliziert. Damit erhält man die tatsächliche Verkleinerung des Bildes. Das ist der erste Faktor der finalen Formel.

Zusätzlich muss die durch den Drehgeber eingelesene Veränderung der Bildgröße berücksichtigt werden. Diese Veränderung G ist der prozentuale Teil, den die Pixel-Breite P_{bild} des zu verarbeitenden Bildes von den 854 Pixeln Bildbreite P_{beamer} des Projektors einnimmt.

$$G = \frac{P_{bild}}{P_{beamer}}$$

Die finale Breite P_{final} des Bildes berechnet sich aus der Kombination der zuvor berechneten Teilschritte.

$$P_{final} = P_{bild} * (d_{max} - d_{sensor} * f) * G$$

Die Formel lässt sich im Grunde auf einen kalibrierbaren Faktor herunterbrechen. Das ist die Messung der Distanz im Verhältnis der Breite des Bildes.

```

200 #Prozent des Beamerdisplays
201 percent_width = current_width / pixel_width_beamer
202
203 #Berechnung der abzuziehenden Pixel
204 distance_value = (dist_a4 - dist) * lense_factor
205 pixels_per_cm = width_a4 / cm_a4 #118 px/cm
206 pixels_to_subtract = pixels_per_cm * distance_value
207 distance_factor = (width_a4 - pixels_to_subtract) / width_a4
208
209 #Verrechnung der Faktoren
210 new_width = width_a4 * percent_width * distance_factor

```

7.4.5 Linienfinder bei Handyfotos

Der Linienfinder oder auch Linienextraktor, ist dazu da, die Linien eines Bildes herauszufiltern. Dieser Filter ist beispielsweise dann vorteilhaft, wenn man eine analoge Zeichnung auf Papier digitalisieren möchte. Fotografiert man die Zeichnung beispielsweise mit dem Handy und schickt diese anschließend durch den Linienextraktor, dann bleiben nur noch die eigentlichen Linien der Zeichnung übrig, der Rest wird mit weiß aufgefüllt. Der Filter arbeitet linear. Er kontrolliert jeden Pixel des Bildes auf seine Farbwerte. Ist der Pixel heller als ein bestimmter Schwellenwert, wird der Pixel weiß gefärbt. So wird das komplette Bild abgearbeitet. Anschließend werden die dadurch entstandenen harten Kanten des Bildes leicht abgeschwächt, indem die Farbe der umliegenden Pixel mit der Farbe des Pixels verrechnet wird. So bleiben nach der Filterung nur die Linien, also die dunklen Bildbereiche bestehen.

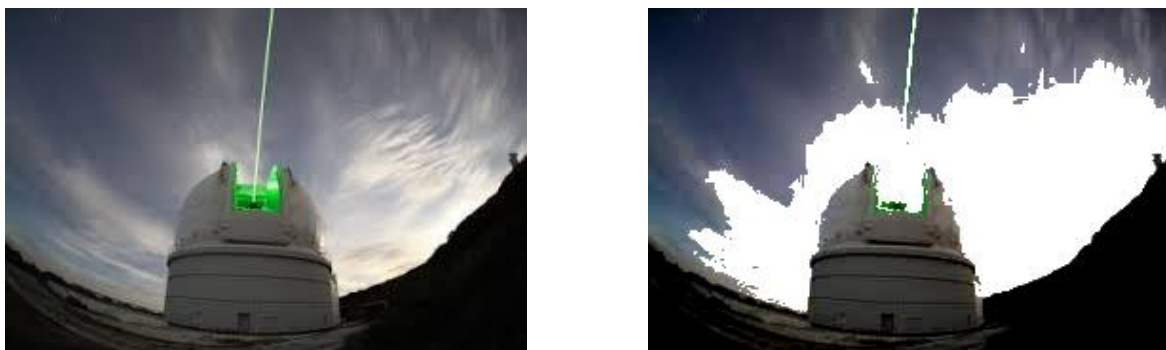


Abbildung 12: Beispielbild vor und nach der Verarbeitung

7.4.6 Senden des Druckauftrags

Das Senden des Druckauftrages stellt den letzten Schritt in der in der technischen Spezifikation geplanten Datenpipeline dar. Das final berechnete Bild muss nun an den gewünschten Drucker gesendet werden. Zum Senden eines normalen Druckauftrags werden normalerweise nur die Mac Adresse des Users und der dazugehörige Channel benötigt. Allerdings ist der in diesem Fall verwendete Drucker kein normaler Tintendrucker, sondern ein Stencil-drucker. Damit der Drucker dazu in der Lage ist die gesendete Bilddatei auf das Transferpapier zu übertragen, müssen deshalb einige vom Hersteller bereitgestellte Pakete installiert werden. Leider sind die vom japanischen Hersteller Brother zur Verfügung gestellten Pakete weder gut gewartet noch gut für ein Raspberry Betriebssystem nutzbar. Der Hersteller gibt sogar auf seiner Seite an, die Installation des Paketes mit der fragwürdigen Flag *force-all* durchzuführen, die jeden aufkommenden Fehler unterdrückt.

Die ersten Testdrucke, die den Weg durch den Treiber zum Drucker schafften, weisen eine streifenartige Struktur auf. Die Vermutung liegt nahe, dass diese durch den Druck der Druckerwalze auf dem Stencil-Papier entstanden sind und die vom Drucker empfangene Datei leer war. Doch schaut man sich die entstandenen Streifen genauer an, erkennt man einzelne Zeichen und Buchstaben in den Zeilen.

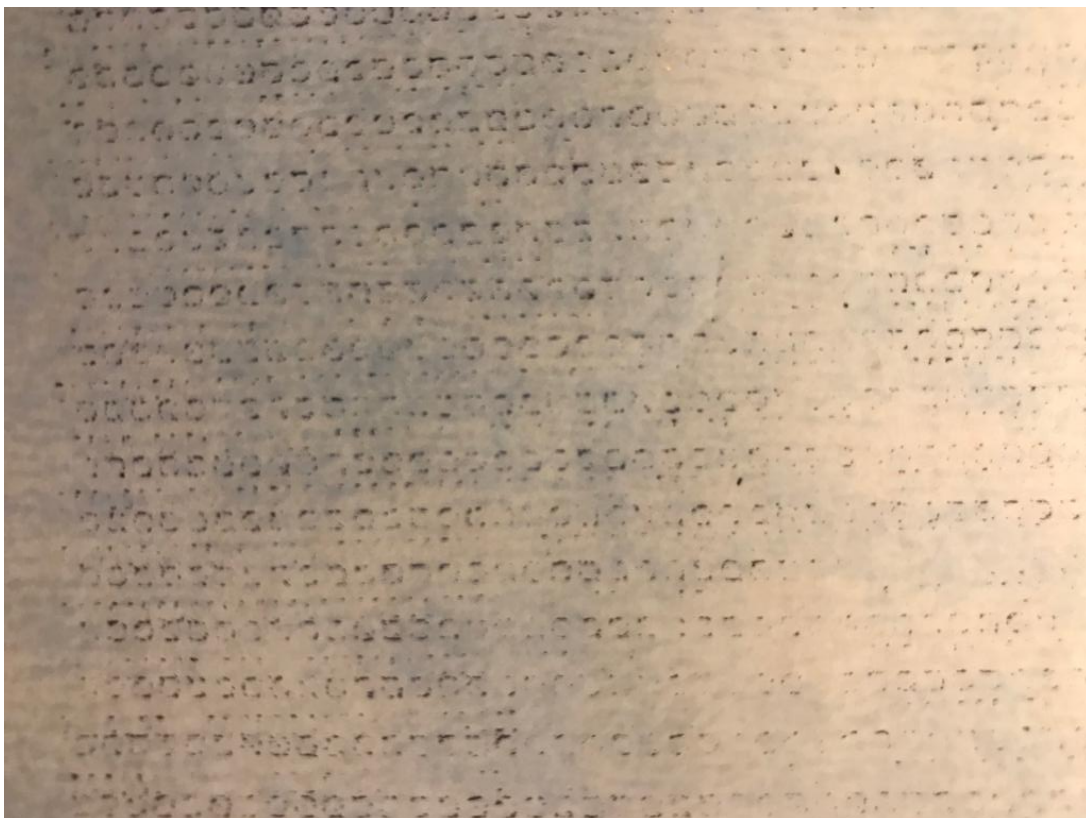


Abbildung 13: Erkennbare Zeichen auf dem gedruckten Papier

Der Drucker hat also eine Datei empfangen, diese jedoch falsch interpretiert. Statt eines Bildes hat dieser die Datei als Text und Zeichen interpretiert. Damit liegt das Problem nicht beim Drucker, sondern vielmehr im Drucktreiber des Herstellers auf dem Raspberry Pi. Bisher wurde der Druckauftrag mit dem Konsolen Befehl `lp` abgeschickt, doch möglicherweise muss der Auftrag direkt an den Druckertreiber gesendet werden, nicht an den Drucker.

Das Programmieren einer Schnittstelle zwischen dem Raspberry Os und dem Druckertreiber ist jedoch kein Teil dieser Arbeit. Als Umgehung des Problems druckt der Prototyp das Bild nicht mehr selbst aus. Er stellt dem User, sobald das finale Bild berechnet wurde, einen Download-Button zur Verfügung. Mit diesem kann man das Bild direkt von der Website herunterladen. Glücklicherweise konnte der Druckauftrag nach der Installation der nötigen Treiber von einem Windowsgerät deutlich einfacher abgesendet werden. Außerdem kann das finale Bild so auf verschiedenen Druckern gedruckt werden, falls der User das wünscht.

7.5 Gehäuse

Um das Gehäuse des Prototyps im Programm *FreeCAD* für den Druck vorzubereiten, muss die Elektronik auf den Millimeter genau ausgemessen werden. Es muss darauf geachtet werden, an welcher Stelle welcher Sensor und welcher Knopf seinen Platz findet. Es müssen Löcher im Gehäuse für die Stromversorgung eingeplant werden. Dieses Verfahren stellt sich als sehr komplex und zeitaufwändig heraus. Das erstellte 3D Modell weicht deshalb vom der im Grafikprogramm erstellten Konstruktionsskizze ab. Der erste Druck des Gehäuses dauert 10 Stunden und 30 Minuten. Anschließend muss das Stützmaterial entfernt werden. Der erste Versuch des Drucks weist leider einige Probleme auf. Der Platz für die Projektorplatine ist viel zu klein und die Kabel stehen hervor. Im zweiten Entwurf werden diese Probleme berücksichtigt und das finale Gehäuse kann gedruckt werden.

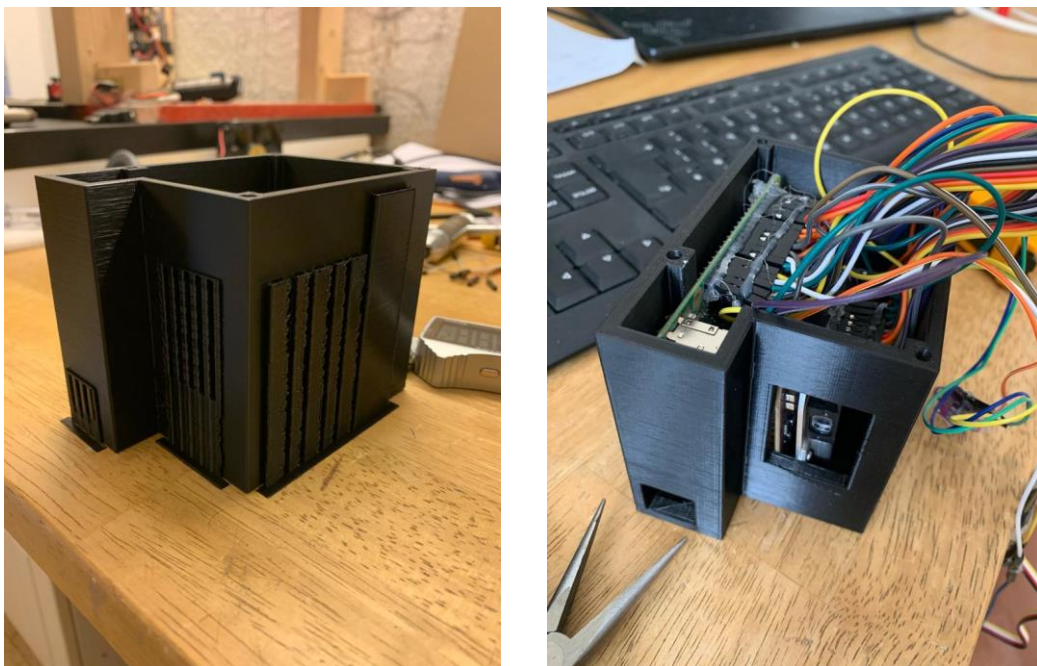


Abbildung 14: Das erste Gehäuse des Prototyps

8 Evaluation

Um die Funktionen des Prototyps in verschiedenen Bereichen zu überprüfen, wurden drei verschiedene Testreihen durchgeführt. Diese sollen einen Überblick über den generellen Nutzen des Gerätes geben und nötige Schritte zu einem fertigen Produkt aufzeigen.

8.1 Testreihe

8.1.1 Effizienz der Filter



Abbildung 15: Testreihe zur Effizienz der Filter

8.1.2 Projektion auf verschiedenen Körperteilen



Abbildung 16: Testreihe zur Projektion auf verschiedene Körperteile

8.1.3 Größenberechnung des finalen Drucks

Entfernung in cm	Abweichung in mm
30	0
35	0
40	1
45	1
50	6
60	10

Tabelle 4: Messung des absoluten Fehlers

8.2 Gütemaß

In der ersten Testreihe wurde ersichtlich, dass die programmierten Filter die Sichtbarkeit der Bilder stark verbessern. Die Linien des Motivs sind so deutlich genauer erkennbar.

Die Projektion auf weniger gewölbte Körperteile funktioniert deutlich überzeugender als die Projektion auf beispielsweise einen Arm. Diese Tatsache hängt aber auch von der Distanz zum Körperteil ab.

An den Messungen kann man erkennen, dass die Werte auf mittlere Distanz sehr geringe Abweichungen aufweisen. Auf weitere Distanzen ab 50 cm weicht die Größe der gedruckten Bilder etwas stärker vom projizierten Bild ab. Da der Prototyp voraussichtlich meist in mittlerer Distanz verwendet werden wird, ist die Abweichung des gedruckten Bildes zu vernachlässigen. Ein Grund für die Abweichung auf größere Distanzen kann die Beschaffenheit des Time-of-Flight Sensors sein. Die Werte dieses Sensors schwanken mehr und mehr, je größer die Distanz wird. Dadurch entsteht ein systematischer Fehler. Eine zufällige Fehlerquelle kann die Kalibrierung der Software darstellen oder menschliches Versagen in der Messung von Abstand und Größe.



Abbildung 17: Messung des absoluten Fehlers

Der Prototyp liefert sehr zufriedenstellende Ergebnisse. Auf größere Distanzen tritt eine leichte Abweichung des gedruckten Bildes auf. Um den Fehlergrund genau zu lokalisieren wären weitere Tests auf unterschiedlichen Distanzen notwendig. Um den Prototyp zu einem fertigen Produkt weiterzuentwickeln, sollte die Abweichung der Größe auf maximal einen Millimeter reduziert werden.

8.3 Kostenberechnung

Das für das Tätowierhandwerk verwendete Thermotransferpapier kostet ungefähr 47 Euro für 100 Blatt. Ohne den Prototypen muss man das Motiv ungefähr drei Mal ausdrucken, bis die richtige Größe gefunden wurde. Das sind also pro Motiv zwei gesparte Blätter des Thermotransferpapiers, also insgesamt 0,94 Euro. Zusätzlich muss der User für den Druckprozess deutlich weniger Zeit verwenden. Grob geschätzt spart der User mit Verwendung des Gerätes 5 Minuten. Bei einem geschätzten Stundenlohn eines Tätowierers von 100 Euro ergibt das ein Ersparnis von 8,30 Euro.

Mit Verwendung des Prototyps kann ein User also bis zu 9,24 Euro pro Kunde sparen. Bei geschätzten 5 Kunden am Tag addiert sich diese Ersparnis auf 46,2 Euro am Tag. Natürlich ist diese Zahl sehr stark vom User und dessen Umsatz und Arbeitsgeschwindigkeit abhängig. In Tätowier-Studios mit mehreren Leuten kann sich dieses Ersparnis noch multiplizieren.

9 Zusammenfassung

In dieser Arbeit wurde im Ersten Schritt ein Plan für den Bau eines Prototyps entwickelt. Anschließend folgte das Verkabeln und Testen der Teile und Sensorik des Gerätes entsprechend der Planung. Danach wurde eine auf den Kunden zugeschnittene Oberfläche implementiert und mit zusätzlichen Konfigurationen für die Bilddatei versehen. Als nächstes wurde die Firmware des Prototyps programmiert, welche die Bilder darstellt und manipuliert. Es wurden verschiedene Filter zur Verbesserung des Ergebnisses programmiert. Anschließend wurde eine Funktion zur Glättung der Sensordaten geschrieben und das Ergebnis mit der Bilddatei verrechnet. Die Methoden der Firmware wurden mehrmals optimiert, um die Performance des Prototyps zu maximieren. Schließlich wurde ein Gehäuse für das Gerät entworfen und anschließend mit einem 3D Programm konstruiert und ausgedruckt. Die Funktionsweise des Prototyps wurde auf seine Güte getestet und seine Nützlichkeit evaluiert.

Die größten Hürden bei der Entwicklung stellte die Verkabelung und die Datenübertragung zwischen den Komponenten dar. Es gibt teilweise so viele mögliche Fehlerquellen, dass es sehr schwierig sein kann einen Fehler zu finden. Eine weitere Herausforderung stellte die finale Berechnung der Bildgröße dar.

Dennoch ist das Ergebnis der Evaluation sehr positiv. Das Gerät kann den Alltag der Tätowiererin definitiv vereinfachen und Kosten sparen.

10 Ausblick

Jeder fünfte Deutsche ist laut Aussagen einer Umfrage des Ipsos Instituts tätowiert. Die große Nachfrage der Bevölkerung wird von etlichen Tätowier-Studios in jedem Teil des Landes befriedigt. Es ist abzusehen, dass dieser Marktzweig sich in den folgenden Jahren verbreitert und modernisiert. Eine digitale Neuerung, wie der in dieser Arbeit entwickelte Prototyp, kann auf diesem fruchtbaren Boden für eine ganze Reihe Neuentwicklungen und Inspirationen sorgen. Schafft man es, sowohl das Image des Produktes als auch seine Handhabung perfekt an die Bedürfnisse der Studios anzupassen, kann die Idee sogar zu einer Unternehmensgründung führen. Man darf dabei nicht vergessen, dass der Anwendungsbereich des Prototyps über die Tätowier-Studios hinausreicht. So wäre es beispielsweise auch denkbar, dass ein Maler das Gerät für seine Arbeit als Hilfe für die Proportionen seiner Werke nutzt. Ein Kachel-Leger könnte mit dem Prototyp die Genaue Anzahl Kacheln für eine Wand in kürzester Zeit kalkulieren. Man könnte die perfekte Größe für ein Plakat an einer Wand direkt messen, ohne den Zollstock zur Hand zu nehmen. Die Anwendungsszenarien sind mannigfaltig.

10.1 Kalibrierung

Der Prototyp kann durch einen einfachen Schritt vom User kalibriert werden. Die Formel zur Berechnung der finalen Bildgröße zeigt, dass maximale Distanz für die Kalkulation des Linsenfaktors und damit der gesamten Größenberechnung entscheidend ist. Der User könnte also in einem Kalibrierungsschritt dazu aufgefordert werden, den Prototyp so zu halten, dass die Projektion genau die Breite eines A4 Blattes aufweist. Dazu könnte der User das Blatt auch auf den Boden legen. Anschließend wird die Distanz zu dem A4 Blatt gemessen. Daraus können anschließend alle weiteren Werte berechnet werden und die Kalibrierung ist beendet.

10.2 Mögliche Verbesserungen

Um den Prototypen zu einem marktfähigen Produkt weiterentwickeln zu können, müssen einige Vorgänge optimiert werden. Die Web-Oberfläche sollte einen moderneren Look bekommen. Es wäre außerdem sinnvoll die Idee des emulierten Druckers in die Tat umzusetzen. Zudem sollte das gedruckte Bild auch bei größeren Abständen nur sehr wenig vom projizierten Bild abweichen. Abschließend sollte die Stromversorgung nicht mehr über zwei verschiedene Kabel erfolgen, sondern entweder über Akku oder maximal ein Kabel.

10.3 Persönliche Meinung

Es gibt eine sinkende Zahl an Tätowierern, die frei Hand arbeiten. Leider finden sich zu diesen Themen wenige verlässliche Informationen, da viele Bereiche der Tätowier-Branche schlecht dokumentiert sind. Ich bin mir sicher, dass die jüngere Generation der Tätowierer eher zum Tablet greifen wird, um das Tattoo zu designen als die ältere Generation. Daher denke ich, dass der Prototyp eine große Hilfe für ein Tätowier-Studio sein kann.

11 Quellennachweis

- Cliff Ortmeyer (2014). *A Brief History of Single Board Computers* URL: <https://canada.newark.com/wcsstore/ExtendedSitesCatalogAssetStore/cms/asset/pdf/ame-ricas/common/NE14-ElectronicDesignUncovered-Dec14.pdf> (besucht am 20.05.2022)
- Banzi, Shiloh (2022). *Getting started with Arduino* ISBN: 9781449363338
- (o.D.[a]). URL: <https://www.heise.de/preisvergleich/raspberry-pi-4-modell-b-a2081132.html> (besucht am 20.05.2022)
- Martin Belam (2012). *The Raspberry Pi: reviving the lost art of children's computer programming*. In: the Guardian. Guardian News and Media Limited, 29. Februar 2012,
- (o.D.[b]). *Raspberry Pi Model Specifications* (2014). URL: <https://web.archive.org/web/20140924025606/http://www.raspberrypi.org/documentation/hardware/raspberrypi/models/specs.md> archiviert am 24.09.2014, (besucht am 21.05.2022)
- Brian Behlendorf (2012). *The Apache Software Foundation*, In: Computer. Volume 45, Issue 10
- Nicolas La Rocco (2021). *Apple iPhone 13 Pro Max im Test*, In: Computer Base, URL <https://www.computerbase.de/2021-10/apple-iphone-13-pro-max-test/2/> (besucht am 30.05.2022)
- John Miano (1999). *Compressed image file formats*, ISBN: 978-0201604436
- J van der Knijff (2009). Unventory of long-term preservation risks URL: http://opf-labs.org/format-corpus/pdfCabinetOfHorrors/test_fontArialNotEmbedded.pdf (besucht am 02.06.2022)
- Samarth Brahmhatt (2013). *Practical OpenCV* ISBN: 9781430260806
- Stefan Van der Walt (2014). *image processing in python* In: PeerJ (besucht am 02.06.2022)
- Alex Clark (2019). *Pillow (pil fork) documentation* URL: <https://www.realmoon.net/wordpress/wp-content/uploads/2019/07/pillow.pdf> (besucht am 25.06.2022)
- (o.D.[c]). *IPP-Protokoll* URL: <https://www.itwissen.info/IPP-Protokoll-Internet-printing-protocol-IPP.html> (besucht am 13.06.2022)
- David Kalinsky, Roee Kalinsky (2009). *Introduction to I2C* URL: <https://infocon.org/cons/DEF%20CON/DEF%20CON%2017/DEF%20CON%2017%20badge/Other%20Stuff/Introduction%20to%20I2C%20-%20Embedded.com.pdf> (besucht am 20.06.2022)
- Alex Karpinsky (2017). *The fastest production-ready image resize out there, part 0*, <https://uploadcare.com/blog/the-fastest-image-resize/> (besucht am 20.06.2022)
- (o.D.[d]). *I2C Bus* URL: <https://evision-webshop.de/i2c-debugging> (besucht am 21.06.22)

12 Abbildungsverzeichnis

Abbildung 1: Raspberry Pi 3 - Platine	10
Abbildung 2: Arbeitsweise eines Time-of-Flight-Sensors	11
Abbildung 3: Master-Slave System des I2C-Protokolls	14
Abbildung 4: Zustandsdiagramm des Prototyps.....	15
Abbildung 5: Lesbarkeit der Bildbreite	16
Abbildung 6: Software-Schichten des Prototyps	19
Abbildung 7: Visuelle Erscheinung mit Seiten- und Draufsicht.....	20
Abbildung 8: Darstellung des I2C-Ports 0x29	23
Abbildung 9: Elektronik der Projektorplatine	23
Abbildung 10: Angezeigte Fehlermeldung zur Niedrigspannung.....	24
Abbildung 11: Web-Oberfläche des Prototyps	26
Abbildung 12: Beispielbild vor und nach der Verarbeitung	31
Abbildung 13: Erkennbare Zeichen auf dem gedruckten Papier	32
Abbildung 14: Das erste Gehäuse des Prototyps	33
Abbildung 15: Testreihe zur Effizienz der Filter.....	34
Abbildung 16: Testreihe zur Projektion auf verschiedene Körperteile	34
Abbildung 17: Messung des absoluten Fehlers	35

13 Tabellenverzeichnis

Tabelle 1: Die Bauteile und ihre Aufgabenbereiche	18
Tabelle 2: Die Programmiersprachen und ihre Aufgabenbereiche	19
Tabelle 3: Manuelle Messung von Distanz und Breite des Bildes	29
Tabelle 4: Messung des absoluten Fehlers	35