CARL VON OSSIETZKY universität OLDENBURG

Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Bachelorstudiengang Informatik

Bachelorarbeit

BeadBox: Entwicklung eines informatischen Systems zur automatisierten Erstellung von Bügelperlenbildern

vorgelegt von:
Henning Ziegler

Gutachter:
Prof. Dr.-Ing. Oliver Theel
Robert Schadek, M.Sc. Inform.

Oldenburg den, 11. September 2013

Informatische Systeme sind in der heutigen Zeit nicht mehr weg zu denken. Oft ist deren Funktionsweise jedoch komplex und nur schwer verständlich. Gerade auf Informationsveranstaltungen für Schüler ist dies ein Problem, da es an Möglichkeiten fehlt den Schülern diese Fähigkeiten auf anschauliche Weise zu demonstrieren. In dieser Arbeit wird ein System entwickelt, welches die Fähigkeiten informatischer Systeme auf anschauliche Weise demonstriert und dadurch das Interesse an der Informatik wecken hilft. Das System, mit dem Namen *BeadBox*, erstellt aus einem beliebigen Ausgangsbild ein entsprechendes Bild aus Bügelperlen. Bei der *BeadBox* kann man das Zusammenspiel zwischen Hard- und Software und die Sicherstellung des Fertigungsprozesses durch Fehlererkennungsverfahren beobachten. Diese Arbeit geht auf die bei der Entwicklung zu lösenden Probleme sowie den Entwurf der Hardware und der Software, die das System *BeadBox* bilden, ein.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Hintergrund

Bügelperlen Bügelperlen sind kleine, in mehreren Farben erhältliche Plastikröhrchen für Bastelarbeiten. Es gibt sie in mehreren Größen: Mini mit einem Durchmesser von 2,5 mm, Midi mit einem Durchmesser von 5 mm und Maxi mit einem Durchmesser von 10 mm. Die Perlen werden auf Stiftplatten gesteckt, um so Bilder oder Muster zu erstellen. Alternativ kann man sie auch fädeln und damit Ketten oder Armbänder herstellen. Durch Bügeln verschmelzen die einzelnen Perlen miteinander und bilden ein zusammenhängendes Bild, welches als Dekoration oder als Untersetzer (beispielsweise für Kaffeetassen, etc.) verwendet werden kann.

Die Zielgruppe der Bügelperlen sind in der Regel Kindergarten- und Grundschulkinder ab 3 Jahren. Mit den Bügelperlen können leicht und schnell greifbare Ergebnisse erzielt und gleichzeitig die motorischen und kreativen Fähigkeiten trainiert werden.

Die Bügelperlen, auch Hama-Perlen genannt, werden seit 1971 von der Malte Haaning Plastic A/S (Dänemark) vertrieben. Die Malte Haaning Plastic A/S wurde 1961 von Malte Haaning gegründet und stellte anfangs Trinkhalme sowie später weitere Kunststoffartikel her. Seit 1984 ist Hama (aus den ersten beiden Buchstaben des Nachnamens und den ersten beiden Buchstaben des Vornamens zusammengesetzt) eingetragenes Warenzeichen [1].

Namensherkunft Der Name *BeadBox* setzt sich aus dem englischen Wort für Bügelperlen *bead* und dem englischen Wort für Kasten/Gehäuse *box* zusammen. Mit dem Wort *box* soll verdeutlicht werden, dass es sich um ein geschlossenes, eigenständiges System handelt.

1.2. Aufgabenstellung

Das Thema dieser Arbeit ist die Entwicklung der *BeadBox*, eines informatischen Systems (d.h. Hard- und Software) zur automatisierten Erstellung von Bügelperlenbildern. Das System soll in der Lage sein anhand eines beliebigen Bildes (Portrait, Logo, etc.), ein (grob) gerastertes und an die vorhandenen Perlenfarben angepasstes Bügelperlenbild zu erstellen.

Diese Arbeit beinhaltet die Konzeption und den Bau der Maschine, die für den Setzvorgang zuständig ist, sowie die Entwicklung der Steuerungssoftware, die die Benutzerführung, die Bildverarbeitung und den Steuerungsvorgang übernimmt. Die Maschine muss dabei die Bügelperlen korrekt auf die entsprechende Stelle der Stiftplatte setzen können. Die Software soll dafür das Bild rastern und an die vorhandenen Farben anpassen und anschließend den Setzvorgang steuern. Dabei muss die Synchronisation zwischen Hard- und Software gewährleistet sein.

Das System soll beispielsweise auf Informationsveranstaltungen für Schüler eingesetzt werden und dort das Interesse und den Spaß an der Informatik, durch die anschauliche Demonstration der Fähigkeiten informatischer Systeme und das Zusammenspiel

von Hard- und Software, wecken.

Die Gliederung dieses Dokuments entspricht der Reihenfolge des Vorgehens bei der Entwicklung der *BeadBox*. Zuerst werden der Systementwurf und die aus der Aufgabestellung und der Problemanalyse gesammelten Anforderungen diskutiert. Anschließend werden die verwendeten Technologien beschrieben. Danach wird der konkrete Aufbau der einzelnen Hardwarekomponenten und deren Funktion erklärt. Daraufhin wird ein Blick auf die Steuerungssoftware geworfen. Dazu wird u.a. auf die Kommunikation mit der Hardware (Synchronisation, Fehlertoleranz), die Bildverarbeitung (Rastern, Farbanpassung) und die Setzverfahren (zeilenweise, farbweise) eingegangen. Nachdem die Einrichtung und die Inbetriebnahme der *BeadBox* beschrieben wurde, wird die Zuverlässigkeit und die Performanz des Systems anhand einiger Tests und Benchmarks untersucht. Abschließend wird ein Fazit der Entwicklung gezogen und ein Ausblick auf mögliche Verbesserungen und Erweiterungen gegeben.

2. Entwurf

In diesem Abschnitt werden die aus der Aufgabenstellung resultierenden Anforderungen sowie der Systementwurf behandelt.

2.1. Problemanalyse

Mit der *BeadBox* soll es möglich sein, aus einem beliebigen Bild ein Bügelperlenbild zu erstellen. Dazu müssen im Wesentlichen vier Probleme gelöst werden:

1. Das Ausgangsbild muss an das Raster der Stiftplatte angepasst werden.
2. Für die Ausgabe des Bildes müssen verschiedenfarbige Bügelperlen vorhanden sein.
3. Die Bügelperlen müssen korrekt auf die Stifte der Stiftplatte abgesetzt werden. Hierzu ist eine genaue xy-Positionierung erforderlich.
4. Es darf immer nur genau eine Perle senkrecht pro Stift abgesetzt werden. Die Perlen müssen daher ausgerichtet und vereinzelt werden.

Das Ausgangsbild besteht in der Regel aus einer recht großen Anzahl einzelner Bildpunkte (Pixel). Auf der Stiftplatte sind jedoch nur verhältnismäßig wenige Stifte vorhanden, sodass eine Anpassung des Ausgangsbildes an diese geringere Auflösung notwendig ist. Dieser Anpassungsvorgang ist relativ anspruchsvoll, da trotz der geringen Auflösung der Stiftplatte die wesentlichen Bildmerkmale erhalten bleiben sollen. Zudem gibt es nur eine begrenzte Anzahl an verfügbaren Bügelperlenfarben. Deswegen muss eine Anpassung der Ausgangsfarben vorgenommen werden. Als Ergebnis dieses Vorgangs erhält man eine Schablone, die die Perlenfarben und Positionen für den Druckvorgang enthält.

Die Ausgabe des Bildes erfolgt in den folgenden Schritten: Farbe der Perle auswählen → Positionieren → Perle absetzen. Dieses Vorgangsschema wird wiederholt, bis alle Perlen des Bildes gesetzt wurden.

Systeme, die ein solches oder ähnliches Vorgehen implementieren, sind z.B. Tintenstrahldrucker oder Platinenbestückungssysteme.

Bei Tintenstrahldruckern wird die Druckposition durch Verschieben des Druckkopfes zusammen mit den Farbpatronen (x-Achse) und dem Verschieben des Papiers (y-Achse) erreicht. So kann jeder Punkt auf dem Papier angefahren werden. Das Auswählen der Farbe und das „Absetzen“ (also das Drucken der Farbe) erfolgt bei Tintenstrahldruckern allerdings in einem Schritt.

Ein weiteres Beispiel sind automatische Platinenbestückungssysteme, welche bei der massenhaften Bestückung von Platinen mit Bauteilen eingesetzt werden. Die Bauteile werden dabei aus einem Magazin entnommen und mit hoher Geschwindigkeit an der entsprechenden Position auf die Platine gesetzt. Die Bauteile sind in dem Magazin stets exakt ausgerichtet, sodass das Aufnehmen der Bauteile sehr schnell erfolgen kann. Bei der Positionierung wird je nach Typ entweder die Platine oder die Bestückungseinheit bewegt. Oft verwenden diese Systeme auch mehrere Absetzvorrichtungen parallel, um eine höhere Fertigungsgeschwindigkeit zu erreichen (Beispiel eines Bestückungsautomaten:

http://www.youtube.com/watch?v=S8qkaTsr2_o). Bestückungsautomaten sind relativ groß und teuer in der Anschaffung [8].

Da die Bügelperlen im Gegensatz zu den Bauteilen in Bestückungsautomaten unorganisiert vorliegen, ist eine Vereinzelung der Perlen notwendig. Zusätzlich muss dafür gesorgt werden, dass die Bügelperlen korrekt ausgerichtet sind, da die Perlen nur senkrecht auf den Stift gesetzt werden können. Außerdem kann es durch die unregelmäßige Form der Bügelperlen schnell zu Verklemmungen kommen.

Für die Vereinzelung und Ausrichtung von Bauteilen werden in der Industrie oft sogenannte Schwing- oder Vibrationswendelförderer verwendet. Solche Förderer besitzen einen Fördertopf, in den die zu vereinzelnden Bauteile eingefüllt werden. Im Vibrationswendelförderer wird durch Elektromagneten eine Mikrowurfbewegung im Behälter erzeugt. Die Bauteile wandern dadurch an einer Spirale am Rand des Fördertopfes entlang. Durch die Vibration reihen sich die Bauteile hintereinander auf und werden zum Auswurf transportiert. Die Vibration beugt außerdem der Verklemmung der Bauteile vor [17]. Aufgrund ihrer Größe und ihrer Komplexität kann die Vibrationsfördertechnik trotz ihrer Vorteile nicht in der *BeadBox* eingesetzt werden.

Für den Druckvorgang kann man aus dem oben genannten Vorgehensschema drei funktionale Hardwarekomponenten der *BeadBox* herleiten: ein Perlenmagazin, eine Positionierungseinheit und ein Vereinzelungssystem. Bei der Entwicklung der Hardwarekomponenten kann auf die Funktionsprinzipien der Industriemaschinen zurückgegriffen werden.

Damit die Hardware ihren Zweck erfüllen kann, wird eine Steuerungssoftware benötigt. Die Software übernimmt die Kontrolle über die einzelnen Hardwarekomponenten und sorgt für den korrekten Ablauf des „Druckvorgangs“. Das Zusammenspiel der Hardware und der Steuerungssoftware bildet das System der *BeadBox* .

2.2. Abgeleitete Systemanforderungen

Aus der Aufgabenstellung und der Problemanalyse resultieren die folgenden nach Hardware und Software getrennten Anforderungen an das zu realisierende System:

Anforderungen an die Hardware:

- **Genaue Positionierung:** Die in dieser Arbeit verwendeten Bügelperlen haben einen Durchmesser von ca. fünf Millimetern. Das Loch in der Mitte hat einen Durchmesser von nur zweieinhalb Millimetern. Fertigungsbedingt haben nicht alle Perlen exakt die vorgegebene Größe und weichen in der Regel um einige Zehntelmillimeter von den Sollmaßen ab. Um zu gewährleisten, dass die Perlen korrekt gesetzt werden, muss die Positionierung sehr genau sein.

- **Möglichkeit des Farbwechsels:** Bilder bestehen in der Regel aus mehreren Farben. Um ein angemessenes Ergebnis zu erhalten, muss die Hardware die Möglichkeit bieten, verschiedenfarbige Bügelperlen zu verwenden.

- **Große Farbbehälter:** Es wäre wünschenswert, wenn das Farbmagazin große Farbbehälter bietet, da größere Bügelperlenbilder aus mehreren hundert bis tausend Perlen bestehen.

- **Kompakter und robuster Aufbau:** Da die *BeadBox* u.a. für Informationsveranstaltungen gedacht ist, soll die Hardware kompakt und robust gebaut sein, um einen einfachen Transport zu ermöglichen. Außerdem sollen mögliche Fehler (Bsp.: Verklemmungen) bereits durch ein geeignetes Hardwaredesign vermieden werden.

Anforderungen an die Software:

- **Einfache Bedienung:** Um den Fertigungsprozess relativ einfach zu gestalten, soll die *BeadBox* eine einfache Benutzerführung bzw. eine einfache Benutzungsoberfläche besitzen.

- **Bildanpassung:** Da die Anzahl der Bügelperlenfarben sowie die Bildgröße begrenzt ist, müssen entsprechende Verfahren vorgesehen werden, um eine Farb- und Größenanpassung der Ausgangsbilder durchzuführen.

- **Feedback:** Dem Benutzer soll der aktuelle Fortschritt des Fertigungsprozesses angezeigt werden. Außerdem soll der Benutzer über eventuell aufgetretene Fehler unterrichtet werden.

- **Synchronisation zwischen Hard- und Software:** Die Software soll sicherstellen, dass der Zustand der Hardware mit dem internen Zustand der Software übereinstimmt.

- **Erweiterbarkeit:** Die Software soll so gestaltet sein, dass etwaige spätere Erweiterungen möglich sind.

- **Erkennung und Behebung von Fehlern:** Größere Bügelperlenbilder können aus mehreren hundert bis tausend Perlen bestehen. Der Fertigungsprozess soll möglichst ohne Eingriff des Benutzers ablaufen. Dazu sollen Fehler erkannt und wenn möglich selbständig behoben werden. Nur bei kritischen Fehlern (Bsp.: Leeres Farbmagazin) soll der Benutzer eingreifen müssen.

- **Implementation:** Die Anwendungssoftware soll auf einem Raspberry Pi laufen.

2.3. Systementwurf

Aus der Aufgabenstellung und der Problemanalyse geht hervor, dass sich die *BeadBox* aus zwei Hauptkomponenten zusammensetzt: 1. der Drucksystem und 2. der Anwendungssoftware (Abbildung 1).

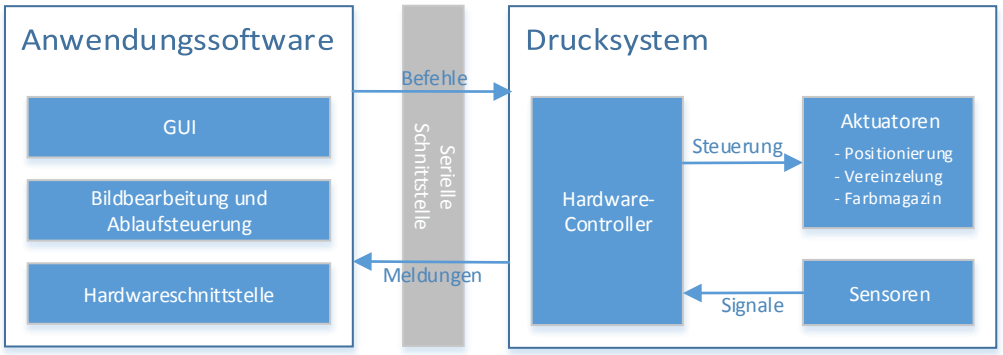


Abbildung 1: Darstellung des Systemaufbaus

Die *Anwendungssoftware* übernimmt die Benutzerführung, die Bildverarbeitung, die Ablaufsteuerung und die Systemüberwachung. Der Anwendungssoftware liegt eine Schichtenarchitektur zu Grunde, wodurch eine gute Erweiterbarkeit und Testbarkeit gewährleistet wird. Die unterste Schicht ist für die transparente Kommunikation mit dem Drucksystem (Hardware-Abstraktion) zuständig. Die darüber angeordneten Schichten implementieren die Bildverarbeitung, die Ablaufsteuerung und die Benutzungsoberfläche. Die Steuerung der *BeadBox* erfolgt über eine übersichtliche Touch-Benutzeroberfläche, um die Demonstration und die Bedienung zu vereinfachen (Abschitt 5). Für die Implementation der Anwendungssoftware wird das Qt-Framework verwendet (siehe Abschnitt 3.3).

Das *Drucksystem* besteht aus vier funktionalen Komponenten: der Positionierungseinheit, dem Vereinzelungsmechanismus, dem Farbmagazin und dem Hardware-Controller.

Die *Positionierungseinheit* dient der Positionierung der Stiftplatte und des Druckkopfes, sodass anschließend eine Perle auf einen Stift abgesetzt werden kann. Diese Positionierung muss sehr genau sein, damit die Perlen exakt auf den Stiften abgesetzt werden können. Die hierfür erforderliche xy-Positionierung wird mit derselben Technik realisiert, wie sie bei CNC- bzw. Portalfräsen eingesetzt wird. Näheres zur Funktion und zum Aufbau der Positionierungseinheit wird in Abschnitt 4.3 erläutert.

Der *Vereinzelungsmechnismus* übernimmt das Absetzen und Vereinzeln einer Perle. Hierbei ist es vor allem wichtig, jeweils nur eine Perle abzusetzen, etwaige Fehler zu erkennen und wenn möglich zu beheben. Da die Perlen fertigungsbedingte Größenunterschiede aufweisen, muss der Vereinzelungsmechnismus mit gewissen Toleranzen arbeiten können. Die Entwicklung dieses Mechanismus wird in Abschnitt 4.4 beschrieben.

Die letzte Einheit ist das *Farbmagazin*. Das Farbmagazin enthält die Farbbehälter für die verschiedenfarbigen Perlen. Das Magazin ist einem Revolver-Magazin nachempfunden, d.h. die Farbbehälter sind in einem Kreis angeordnet, sodass jeder Farbbehälter über den Vereinzelungsmechanismus gedreht werden kann. Weiteres dazu in Abschnitt 4.5.

Zur Steuerung und Überwachung der Hardwarekomponenten wird ein *Hardware-Controller* eingesetzt. Hierfür wird ein Arduino Leonardo verwendet (siehe Abschnitt 3.2).

Auf dem Leonardo Board wird die Ansteuerung der Aktuatoren sowie die Abfrage der Sensoren realisiert. Die Arduino Boards sind für diese Aufgaben gut geeignet, weil sie eine sehr hardwarenahe und trotzdem einfache Programmierung erlauben. Zudem sind sie echtzeitfähig, da sie ohne Betriebssystem funktionieren und so unmittelbar auf Ereignisse reagieren können. Die Controller-Software auf dem Arduino führt die Kommandos der Anwendungssoftware aus und stellt sicher, dass das Drucksystem nicht durch Steuerungsfehler beschädigt wird (Erläuterung in Abschnitt 5.2).

Nachdem nun die grundlegenden Aspekte des *BeadBox*-Entwurfs verdeutlicht wurden, werden im Folgenden die einzelnen Konzepte genauer betrachtet. Es wird zunächst die Konfiguration der verwendeten Entwicklungsumgebung beschrieben. Anschließend wird auf die Entwicklung der Hardware und deren Aufbau eingegangen und danach die Implementierung und die Funktion der Steuerungssoftware erläutert.

2.4. Vorgehensweise

Während der Entwicklung der *BeadBox* war ein strukturiertes Vorgehen sehr wichtig, um eine funktionierende Lösung für die einzelnen Aufgaben zu finden. Bei der Konzeption der Hardware war es besonders wichtig, die Anforderungen an die Steuerungssoftware im Hinterkopf zu behalten. Es musste also auf ein gutes Verhältnis zwischen robustem und fehlertolerantem Aufbau und einer einfachen Steuerung geachtet werden.

Der ursprüngliche Zeitplan sah eine schrittweise Entwicklung der Maschine vor, wobei die Aufgaben nach absteigender Realisierungsschwierigkeit geordnet waren. Nach einer kurzen Zeit der Einarbeitung in die wesentlichen Techniken (CNC-Technik, Schrittmotoren, Arduino, Bügelperlen, etc.) sollte deshalb zunächst der Vereinzelungssmechanismus entworfen und gebaut werden, da dieser potentiell die meisten Probleme verursachen konnte. Danach sollte das Positionierungssystem und abschließend das Farbmagazin gebaut werden.

Da sich jedoch Probleme bei der Realisierung des Vereinzelungssmechanismus ergaben, musste der Aufgabenplan umstrukturiert werden. Um das neue Konzept des Vereinzelungsmechanismus umsetzen zu können, wurde ein Maschinenrahmen samt Positionierungssystem benötigt. Parallel zu den einzelnen Hardwarekomponenten entstanden die notwendigen Module der Steuerungssoftware. Dadurch konnte die Funktion der Hardware schon in einem frühen Entwicklungsstadium getestet werden.

Nach der Erstellung des Konzeptes wurde für jede Hardwarekomponente zunächst ein Prototyp gebaut und manuell getestet. So konnten mit wenig Aufwand verschiedene Lösungen separat ausprobiert und auf ihre Funktion überprüft werden. Außerdem wurden so Fehler bereits früh erkannt und analysiert, sodass sie in der finalen Version behoben werden konnten.

3. Verwendete Technologien

In diesem Abschnitt werden kurz die wesentlichen Technologien (Hard- und Software) eingeführt, die in der *BeadBox* Verwendung finden.

3.1. Raspberry Pi

Der Raspberry Pi ist ein kleiner etwa kreditkartengroßer preisgünstiger Computer. Die Idee für den Raspberry Pi entstand 2006 an der Universität von Cambridge, um den schlechter werdenden Vorkenntnissen der Informatikstudenten entgegen zu wirken. Ziel war es, einen günstigen Computer zum Experimentieren und Programmierenlernen zu schaffen. Das Ergebnis war der Raspberry Pi. Er wird von der Raspberry Pi Foundation entwickelt und vertrieben.

Im Raspberry Pi wird ein SoC (System on Chip) von Broadcom (BCM2835) verwendet, welcher auf der ARM11 Architektur basiert und mit 700MHz taktet. Als GPU dient der ebenfalls von Broadcom hergestellte VideoCore IV, der durch die Unterstützung von OpenGL ES 2.0 in der Lage ist, Videos in Full-HD-Auflösung zu dekodieren und über HDMI abzuspielen. Den Raspberry Pi gibt es in zwei Versionen: das Model A und das überarbeitete Model B. Das Modell A ist mit 265 MB und das Modell B mittlerweile mit 512 MB Arbeitsspeicher ausgestattet. Als Anschlüsse stehen FBAS und HDMI für Video, 3,5 mm Klinkenstecker für Audio, zwei USB 2.0 Anschlüsse (bei Modell A nur einen USB 2.0) und einen 100 MBit Ethernetanschluss zur Verfügung. Weiterhin gibt es 16 sogenannte GPIO-Pins(General Purpose Input/Output), über die zum Beispiel LEDs, Sensoren, Motoren usw. angesteuert werden können. Der Raspberry Pi verfügt außerdem über einen SD-Kartenleser [2] [21]. In diesem Projekt wird des Modell B des Raspberry Pi eingesetzt.

Für den Raspberry Pi sind mehrere ARM-kompatible Betriebssysteme verfügbar. Empfohlen wird die auf Debian basierende Linux-Distribution Raspbian.

3.2. Arduino

Arduino ist eine open-source Plattform für elektronisches Prototyping, die aus Hard- und Software besteht. Als Hardware dient ein einfaches Board, auf dem ein Mikrocontroller mit analogen und digitalen Ein- und Ausgängen sitzt. Die dazugehörige Entwicklungsumgebung basiert auf der Entwicklungsumgebung der Programmiersprache Processing und bietet eine einfachen Einstieg in die Programmierung.

Mit Ausnahme des Arduino Due, der einen ARM-Prozessor besitzt, haben alle Arduino-Boards einen 8-bit Atmel AVR-Mikrocontroller. Kennzeichnend für alle Boards ist, dass die Mikrocontroller bereits mit einem Bootloader vorprogrammiert sind, sodass ein Programmieren direkt über USB/Serielle Schnittstelle ohne zusätzliches Programmiergerät möglich ist. Zudem bieten alle Boards einige analoge Eingangspins und digitale I/O-Pins, von denen einige auch PWM (*pulse width modulation*) unterstützen [3].

Es gibt unterschiedliche Versionen der Arduino-Boards. In diesem Projekt wird der Arduino Leonardo eingesetzt.

Arduino Leonoardo Auf dem Arduino Leonardo ist ein ATmega32u4 Mikrocontroller verbaut, der mit 16 MHz getaktet wird. Das Board bietet insgesamt 20 digitale I/O-Pins, von denen 7 als PWM-Pins benutzt werden können. Zudem gibt es 12 analoge Eingänge. Für die Programmierung stehen 28 KB Flashspeicher zur Verfügung sowie 2,5 KB SRAM und 1 KB EEPROM. Weiterhin ist im ATmega32u4 Mikrocontroller eine serielle Schnittstelle über USB integriert, weshalb man mit dem Arduino Leonardo direkt über eine serielle Schnittstelle kommunizieren kann [4].

3.3. Qt

Qt (ausgesprochen wie das englische Wort *cute*: süß, pfiffig) ist ein C++-Framework zur Entwicklung von Anwendungen und grafischen Benutzeroberflächen und steht unter der GNU Lesser General Public License (LGPL). Qt ist plattformübergreifend einsetzbar und u.a. für Windows, Mac, Linux und neuerdings auch für Android und iOS verfügbar. Anfangs wurde Qt von der norwegischen Firma Trolltech entwickelt, welche im Jahr 2008 von Nokia übernommen wurde. 2011 verkaufte Nokia die kommerzielle Sparte von Qt an das finnische Softwareunternehmen Digia. Gleichzeitig wurde das *Qt-Project* gegründet, in dem Qt als Open-Source-Community-Projekt weitergeführt werden soll.

Im Kern von Qt arbeitet das sogenannte Meta Object System, das C++ um zusätzliche Features, wie Reflection, Properties und ein Signal-und-Slot-System, erweitert. Dazu wird ein Präprozessor, der Meta Object Compiler (kurz MOC), verwendet.

Die aktuelle Hauptversion ist Qt 5. Neu in Qt 5 ist u.a. QtQuick, mit dem man leichtgewichtige animierte GUIs erstellen kann. Als Basis verwendet QtQuick die vom Qt-Projekt entwickelte deklarative Programmiersprache QML (Qt Modeling Language), die eine nahtlose Integration von JavaScript beinhaltet. Neben den Bibliotheken für die Entwicklung von grafischen Benutzungsoberflächen enthält Qt weitere Pakete, z.B. für den Zugriff auf SQL-Datenbanken, XML-Dateien sowie eine Webkit-Integration.

Ein Beispiel für die Leistungsfähigkeit des Qt-Frameworks ist der KDE-Desktop, der zu großen Teilen auf Qt basiert. Neben KDE verwenden auch viele andere Open-Source-Projekte Qt [6] [7].

4. Drucksystem

In diesem Abschnitt wird der Aufbau des Drucksystems der *BeadBox* erläutert und die Funktion der einzelnen Komponenten beschrieben.

Konstruktionszeichnungen Die Zeichnungen wurden mit dem 3D-Modellierungstool *SketchUp Make 2013* von Trimble Navigation Ltd. erstellt. *SketchUp Make 2013* kann kostenlos unter `www.sketchup.com` heruntergeladen werden. Detaillierte Konstruktionszeichnungen befinden sich im Anhang A. Außerdem befinden sich die SketchUp-Dateien auf der beigelegten CD (Anhang C).

4.1. Genereller Aufbau

Das Drucksystem der *BeadBox* ist für die Positionierung, die Vereinzelung und die Farbauswahl verantwortlich (Abbildung 2). Insgesamt betragen die Maße des Drucksystems 40 cm x 45 cm x 50 cm (BxHxT), wobei die Höhe durch die Farbbehälter bedingt ist. Ohne sie ist die Maschine nur 15 cm statt 45 cm hoch. Durch ihren relativ kompakten Aufbau ist sie gut zu transportieren.

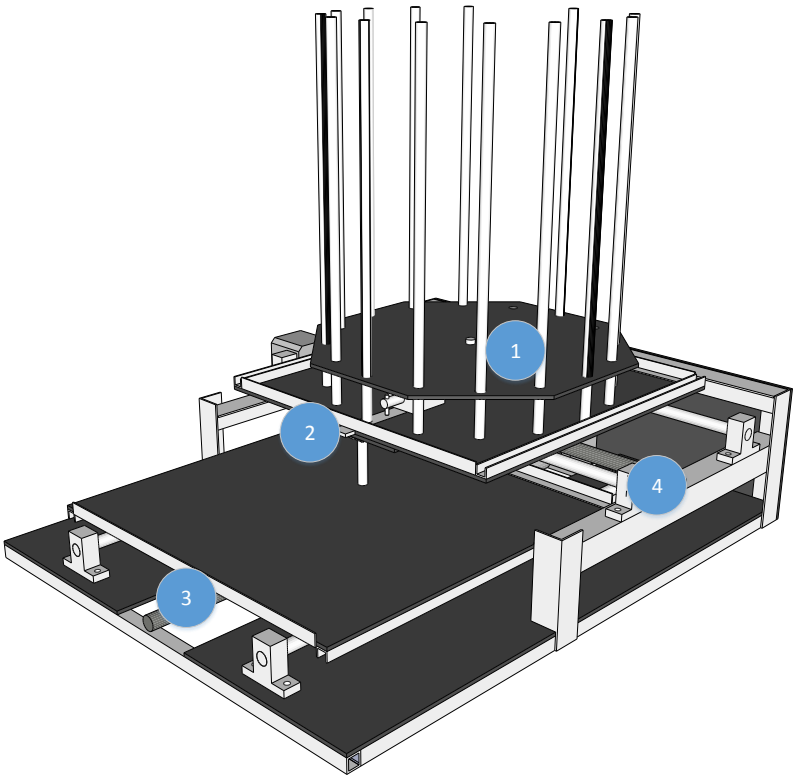


Abbildung 2: Aufbau des Drucksystems der *BeadBox*

Für den Bau der *BeadBox* wurden 3 mm dicke Kunststoffplatten und Aluminium verwendet. Die Bodenplatte sowie der x- und der y-Schlitten sind mit aufgeklebten bzw.

aufgenieteten Aluminiumprofilen verstärkt. Die Baumaterialien wurden so gewählt, dass sie leicht sind, wodurch ein gutes Verhältnis zwischen Gewicht und Stabilität erreicht wurde.

Aus Abbildung 2 ist zu ersehen, dass das Drucksystem aus dem Farbmagazin (1), dem Vereinzelungmechanismus (2) sowie der Positionierungseinheit (3, 4) besteht. Sie ist modular aufgebaut, d.h. die einzelnen Komponenten können ausgetauscht werden. So ist es beispielsweise möglich den Vereinzelungsmechanismus zu wechseln, um eine andere Bügelperlengröße zu verwenden. Auf der Rückseite der Maschine sind die Schrittmotorplatinen, der Arduino und die Verbindungselektronik befestigt.

Anfänglich wurde die Maschine für die Verwendung der Hama-Mini-Perlen mit einem Durchmesser von 2,5 mm konzipiert. Allerdings war dies zu fehleranfällig, weswegen letztendlich doch die Midi-Perlen (Durchmesser 5 mm) benutzt wurden. Durch die von vornherein vorgesehene modulare Bauweise ging der Umbau schnell vonstatten.

4.2. Maschinenrahmen

Der Rahmen der *BeadBox* ist das Grundgerüst auf dem alle weiteren Komponenten aufbauen. Er ist 40 cm x 50 cm groß und 15 cm hoch. In Abbildung 3 ist der Aufbau zu sehen.

Die Grundplatte besteht aus zwei Kunststoffplatten, die auf einen Rahmen aus Aluminiumquadratrohren genietet wurden. Dazu werden gewöhnliche Blindnieten verwendet, die sich hervorragend für die schnelle und feste Verbindung von Bauteilen eignen. Um zwei Bauteile mit Blindnieten zu verbinden, wird zunächst ein Loch in beide Teile gebohrt. Dann wird der Kopf der Blindniete in die Bohrung gesteckt und beide Teile aufeinander gelegt. Nun wird der aus dem Kopf ragende Dorn mit einer Blindnietzange herausgezogen. Dadurch wird der Kopf gestaucht und presst die beiden Bauteile fest zusammen. Sobald die Blindniete festsitzt, wird der Dorn von der Blindnietzange abgetrennt. Die beiden Kunststoffplatten sind links und rechts auf dem Aluminiumrahmen angebracht. In der Mitte der beiden Kunststoffplatten befindet sich eine 8 cm breite Aussparung für die y-Achse der Positionierungseinheit.

An der hinteren Hälfte der Bodenplatte ist zu beiden Seiten ein Aufbau aus Aluminiumwinkeln montiert. Dieser Aufbau ist 9 cm hoch und 30 cm lang. An diesem Aufbau ist die x-Achse der Positionierungseinheit befestigt. Die Winkelkonstruktion erhöht die Steifigkeit des Rahmens und dient außerdem noch als Tragegriffe, an denen das Drucksystem hochgehoben und transportiert werden kann. An der Rückseite ist eine weitere Kunststoffplatte angebracht, die als Befestigung für die Steuerungselektronik dient.

4.3. Positionierungseinheit

Die Positionierungseinheit ist in den Rahmen des Drucksystems integriert (Abbildung 3). Mit ihrer Hilfe kann jeder Punkt auf der Stiftplatte in einem Bereich von 27 cm x 27 cm ansteuert werden. An der Bodenplatte befindet sich die Mechanik für die y-Achse (1). Die Mechanik für die x-Achse (2) ist über der Bodenplatte befestigt. Hierfür

wird dieselbe Technik verwendet, die man auch in CNC-Maschinen einsetzt. Angetrieben werden die Einheiten über je eine Trapezgewindespindel und einem Schrittmotor.

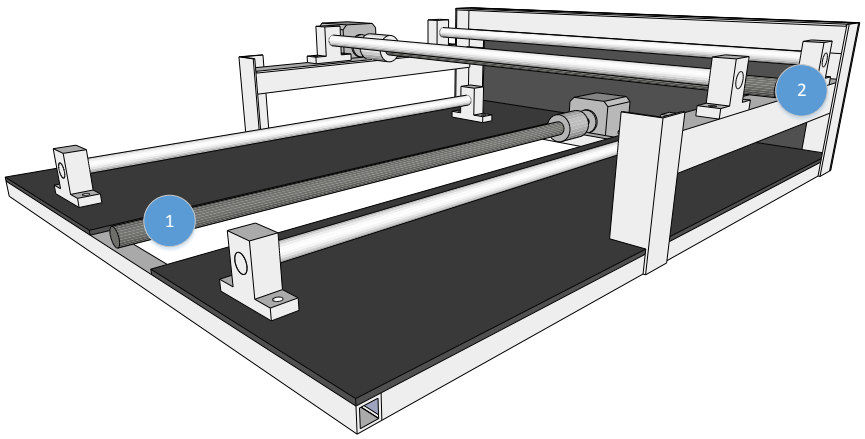


Abbildung 3: Der Rahmen der *BeadBox* mit montiertem Positionierungssystem

Trapezgewindespindeln sind vergleichbar mit normalen Schraubengewinden mit dem Unterschied, dass das Gewinde im Querschnitt eine Trapezform und keine Dreiecksform hat (Abbildung 4). Die Steigung der Trapezgewindespindeln ist zudem genormt. Die in der *BeadBox* verwendete Spindel ist vom Typ TR12x6, d.h. sie hat einen Durchmesser von 12 mm und eine Steigung von 6 mm pro Umdrehung. Auf der Trapezgewindespindel läuft eine passende Trapezgewindemutter, die ohne Spiel auf der Spindel sitzt. In der *BeadBox* werden Muttern aus Kunststoff verwendet, da sie gegenüber Metallmuttern den Vorteil haben, nach einmaliger Schmierung selbstschmierend zu sein. Metallmuttern müssen hingegen in regelmäßigen Abständen gefettet werden, damit sie nicht festsitzen. In teureren CNC-Maschinen werden in der Regel Kugelumlaufspindeln und Kugelumlaufmuttern eingesetzt. Kugelumlaufmuttern haben Ähnlichkeit mit Kugellagern. In der Mutter befinden sich kleine Kugeln, die auf der Spindel laufen. Dadurch ist diese Technik präziser und leichtgängiger, jedoch auch sehr teuer. Teilweise werden auch Zahnriemen eingesetzt. Ein Zahnriemenaufbau wäre allerdings komplexer geworden, da für jede Achse zwei Riemenscheiben als Lager benötigt würden [23].

Die Gewindespindeln werden von Schrittmotoren angetrieben. Bei Schrittmotoren kann die Motorachse schrittweise um einen gewissen Winkel gedreht werden. Sie sind sehr genau und können in der Regel ohne Positionsrückmeldung betrieben werden. Die Schrittmotoren in der *BeadBox* sind bipolare Schrittmotoren (zwei Motorspulen), die insgesamt mit 4 Anschlüsse (zwei pro Spule) ausgestattet sind. Die Motoren haben einen Drehwinkel von 1,8 ° je Vollschritt, d.h. man benötigt 200 Schritte für eine Achsenumdrehung. Die schrittweise Drehung wird durch gezieltes Ansteuern der Motorspulen erreicht. Dabei gibt es zwei Modi, den Vollschritt und den Halbschritt. Im Halbschritt-Modus wird jeder physische Schritt in zwei Schritte unterteilt. Da 200 Schritte allerdings ausreichend genau sind (pro Schritt 0,03 mm Vortrieb), werden die Motoren in der *BeadBox* im Vollschritt betrieben. Tabelle 1 zeigt die Spulenansteuerung für Vollschritte.

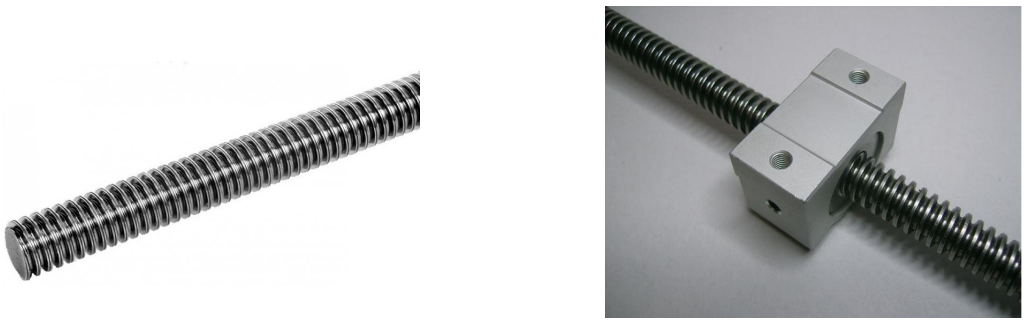


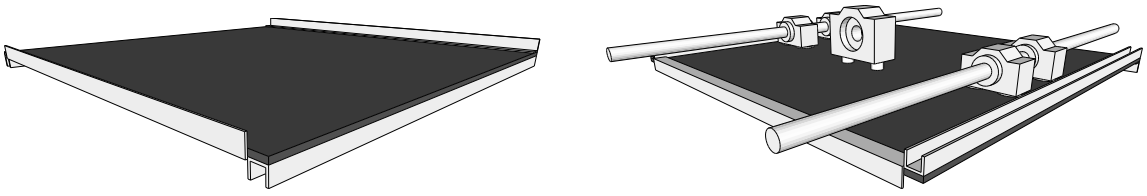
Abbildung 4: Trapezgewindespindel und Trapezgewindemutter,
Quelle: www.dold-mechatronik.de

Bei jedem Schritt sind immer zwei Spulen aktiv. Obwohl Schrittmotoren die Schritte zuverlässig ausführen, kann es bei zu starker Belastung passieren, dass Schritte verloren gehen. Man sollte also darauf achten, dass die Belastung nicht zu stark wird, da sonst Ungenauigkeiten in der Positionierung entstehen [12].

	Phase Spule 1a	Phase Spule 1b	Phase Spule 2a	Phase Spule 2b
Schritt 1	+	-	+	-
Schritt 2	+	-	-	+
Schritt 3	-	+	-	+
Schritt 4	-	+	+	-

Tabelle 1: Bipolare Ansteuerung im Vollschritt

Zu beiden Seiten der Trapezgewindespindeln (1, 2) sind Präzisionswellen aus Edelstahl befestigt. Auf den Präzisionswellen werden die mit Linearlagern gelagerten Schlitten geführt. Der Schlitten für die Stiftplatten ist in Abbildung 5 dargestellt. Er besteht aus einer Kunststoffplatte, die mit aufgeklebten Aluminiumprofilen verstärkt wurde. Von unten sind links und rechts jeweils zwei Linearlager montiert (Abbildung 5b). Durch die zwei beidseitigen Linearlager wird eine Verdrehung vermieden und eine saubere und gerade Führung gewährleistet. In der Mitte der Platte ist die Trapezgewindemutter angebracht.



(a) Ansicht Oben **(b)** Ansicht Unten

Abbildung 5: Schlitten für die Stiftplatten

Der Schlitten für den Druckkopf ist ähnlich aufgebaut, weshalb hier nicht weiter darauf

eingegangen wird. Zu der Positionierungseinheit gehören außerdem noch Anschlagsensoren. Auf jeder Achse gibt es zwei, einen Start- und einen Endsensor. Sie stellen sicher, dass die Schlitten nicht gegen die Wellenlager laufen und die Hardware nicht beschädigt wird. Als Sensoren werden Lichtschranken verwendet. Werden sie unterbrochen, werden die entsprechenden Schrittmotoren sofort abgeschaltet. Außerdem werden die Lichtschranken für die Kalibrierung benutzt, da man durch die Startsensoren eine fest definierte Nullposition erhält.

Ein letztes zu lösendes Problem der Positionierungseinheit war die Stromversorgung des Druckkopfes. Auf dem Druckkopf sind u.a. der Vereinzelungssmechanismus und das Farbmagazin montiert. Beide müssen mit Steuersignalen und Strom versorgt werden. Da sich der Druckkopf auf der x-Achse bewegt, ist keine starre Kabelführung möglich. Auch andere Systeme, wie beispielsweise Drucker und Portalfräsen, haben eine ähnliche Problematik. Als Lösung werden in beiden Systemen oft Kabelschlaufen eingesetzt. Dieses Verfahren wird auch bei der *BeadBox* verwendet. Unter dem x-Achsenschlitten befindet sich ein Flachbandkabel, das so am Schlittenboden befestigt wird, dass eine Schlaufe entsteht. Bewegt sich der Schlitten nun auf der Achse entlang, wird die Schlaufe entweder größer oder kleiner und lineare Bewegungen können ausgeglichen werden.

4.4. Vereinzelungmechanismus

Nachdem eine Position auf der Stiftplatte angefahren wurde, muss eine Bügelperle abgesetzt werden. Dafür sorgt der Vereinzelungsmechanismus, der sich auf dem x-Achsenschlitten befindet. Das Vereinzeln der Perlen gestaltet sich allerdings problematisch. Da die Perlen nur hochkant auf den Stift gesetzt werden können, müssen sie vorher aufgerichtet werden. Außerdem weisen die Bügelperlen fertigungsbedingte Größenunterschiede auf, sodass bei der Entwicklung des Mechanismus gewisse Toleranzen vorgesehen werden müssen. Besonders anwenderfreundlich wäre es, wenn der Mechanismus die Bügelperlen einfach aus einem großen Behälter entnehmen könnte. So entstände kein großer Aufwand beim Nachfüllen der Perlen. Dafür geeignete Verfahren, wie Vibrationswendelförderer, die in der Industrie eingesetzt werden, lassen sich nicht verwenden, da sie zu groß und zu teuer sind. Deswegen musste eine andere Lösung gefunden werden. Insgesamt wurden drei verschiedene Mechanismen ausprobiert, die die oben genannten Probleme lösen sollten.

Erster Lösungsversuch Der erste Lösungsversuch ist in Abbildung 6 zu sehen. Diese Lösung sollte die Vereinzelung und die Ausrichtung der Perlen in einem Schritt durchführen. Der Mechanismus besteht aus einer rotierenden Scheibe, an deren Rand sich ein bügelperlengroßes Loch befindet. Über dieser Scheibe ist ein Perlenbehälter angeordnet. Durch die Rotation sollte eine Perle in das Loch fallen und schließlich abgesetzt werden. Ein ähnliches System wird oft in Tischtennisrobotern zur Vereinzelung der Bälle verwendet[1]. Im Gegensatz zu Tischtennisbällen ist jedoch bei Bügelperlen die Ausrichtung

[1] Joola TT BUDDY: http://shop.joola.de/index.php/zubehor/robots/tt-buddy.html

wichtig. Aus diesem Grund darf das Loch in der Scheibe nur so groß, dass die Perlen nur senkrecht in das Loch passen.

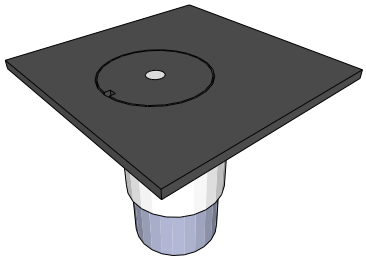


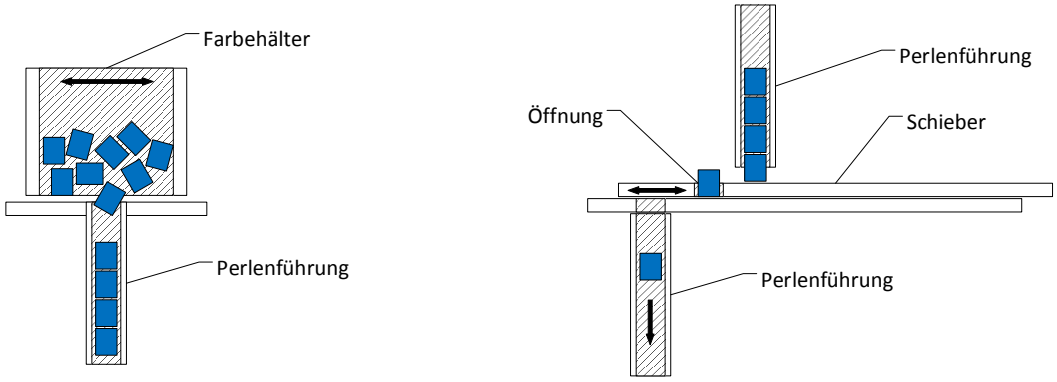
Abbildung 6: Erster Entwurf des Vereinzelungsmechanismus

In den ersten manuellen Tests lieferte dieses System relative gute Ergebnisse. Der erste Prototyp funktionierte jedoch weniger gut. Das Problem war, dass sich die Bügelperlen relativ leicht verklemmten und dadurch den Auslass blockierten. Ebenso oft kam es vor, dass gar keine Perle in das Loch der Scheibe fiel, da sich die Perlen durch die Drehung nicht aufrichteten und daher nicht in das Loch fallen konnten. Eine Vergrößerung des Lochs führte zwar zu einer besseren Sammlung der Perlen, hatte aber zur Folge, dass die Bügelperlen nicht unbedingt hochkant aufgesammelt wurden. Es musste also eine andere Lösung gefunden werden.

Zweiter Lösungsversuch Der zweite Lösungsversuch orientiert sich an dem Konzept der Vibrationswendelförderer. Die Perlen werden in einen runden Behälter geschüttet. Unter dem Behälter befindet sich ein Rohr mit dem Durchmesser der Bügelperlen. Durch eine ruckartige Bewegung des Behälters sollte ähnlich wie bei den Vibrationswendelförderern ein Mikrowurf erzeugt und die Perlen durcheinander geworfen werden. Durch die Bewegung innerhalb des Behälters sollen die Perlen in das Rohr fallen (Abbildung 7a) und so ausgerichtet werden. Unterhalb des Rohrs befindet sich ein Schieber, der die Perlen einzeln aus dem Rohr holen kann. Der Schieber lässt die Perlen dann über dem Stift auf der Stiftplatte fallen (Abbildung 7b). Ein weiteres Rohr dient als Führung, damit die Perlen genau platziert werden können.

An der unteren Perlenführung testet eine Lichtschranke, ob eine Perle abgesetzt wurde. So können mögliche Fehler, wie z.B. ein leerer Farbbehälter, erkannt werden. Für die Rüttelbewegung der Behälter sorgt ein Motor über einen Exzenter. Es werden alle Behälter gleichzeitig bewegt, damit für alle Farben bereits vorausgerichtete Perlen vorhanden sind.

Der Schieber wurde anfangs von einem Servomotor angetrieben. Durch die ständige Belastung ging er jedoch schnell defekt. Daraufhin wurde der Servo durch einen Elektromagnet ersetzt. Ein Elektromagnet ist robuster, da er keine beweglichen Teile besitzt. Beim Anschalten des Elektomagneten wird der Schieber unter die Magazinöffnung gezogen. Im ausgeschalteten Zustand wird der Schieber von einer Metallfeder noch vorne gedrückt. Damit sich die Perlen nicht durch elektrostatische Aufladungen verklemmen können, ist der Schieber aus Aluminium gefertigt.



(a) Vorausrichtung **(b)** Vereinzelung

Abbildung 7: Zweiter Entwurf des Vereinzelungsmechanismus

Finaler Aufbau Mit den Hama-Mini-Perlen (2,5 mm Durchmesser) erzielte die Vorausrichtung durch die Rüttelbewegung gute Ergebnisse. Allerdings traten beim Vereizelungsschritt durch den Metallschieber regelmäßig Probleme auf, weil die kleinen Perlen sehr anfällig für kleinste fertigungsbedingte Ungenauigkeiten sind. Da sich diese Probleme nicht lösen ließen, wurden letztendlich die Hama-Midi-Perlen (5 mm Durchmesser) verwendet. Ihr Vorteil ist, dass sie wegen ihrer Größe weniger anfällig für diese Ungenauigkeiten sind. Ein Nachteil ist jedoch, dass bei den großen Perlen Platzprobleme und oft Verklemmungen bei der Vorausrichtung auftreten, da der ursprüngliche Aufbau für die kleinen Perlen konzipiert wurde. Mit entsprechend größeren Behältern und einem größeren Magazin wäre dieses Problem eventuell lösbar. Dazu hätte jedoch ein Großteil der Maschine neu gebaut werden müssen. Aus diesem Grund wurde nur der Schiebemechanismus weiter verwendet. Anstatt der maschinellen Vorausrichtung, müssen die Perlen nun von Hand in lange Röhren eingefüllt werden.

4.5. Farbmagazin

Die *BeadBox* besitzt ein Farbmagazin, damit verschiedenfarbige Bügelperlen verwendet werden können. Das Farbmagazin befindet sich zusammen mit dem Vereinzelungsmechanismus auf dem x-Achsenschlitten und ist einem Revolvermagazin nachempfunden (Abbildung 8). Die Behälter sind in einem Kreis angeordnet und an einer drehbaren Grundplatte befestigt. Diese Platte kann durch einen Motor um die eigene Achse gedreht werden und so immer einen Behälter über den Vereinzelungsmechanismus positionieren.

Die Farbbehälter (1) sind 30 cm lange Aluminiumröhren mit einem Innendurchmesser von 6 mm. Sie können ca. 60 Bügelperlen aufnehmen. Die Aluminiumröhren sind durch Bohrungen der Magazinplatte gesteckt und mit Kabelschellen und Aluminiumwinkeln befestigt. Die Röhren setzen auf dem Boden des x-Achsenschlittens auf, sodass die Perlen nicht herausfallen können. Am unteren Ende sind die Röhren auf der Vorderseite eingekerbt, damit die Perlen vom Vereinzelungsschieber herausgeholt werden können.

Das Magazin enthält aktuell Behälter für 14 verschiedene Farben. Durch den Motor (2), der sich in der Mitte unter der Platte befindet, können die einzelnen Farbbehäl-

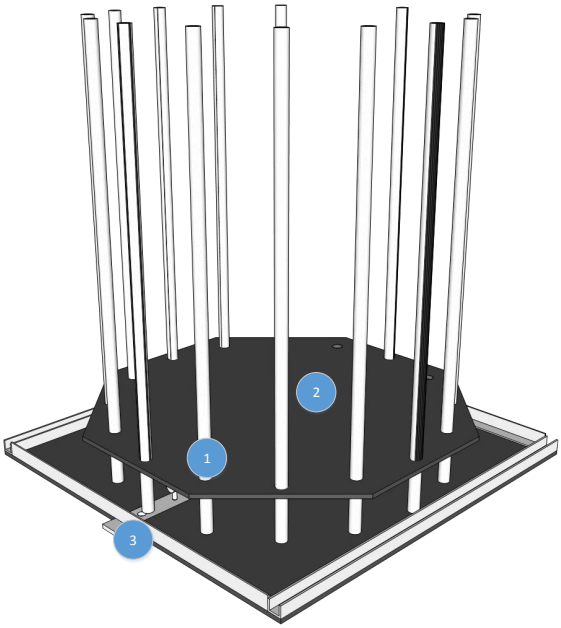


Abbildung 8: Revolver-Farbmagazin

ter über den Vereinzelungsmechanismus (3) gedreht werden. Für jeden Farbbehälter existiert ein Marker, der von einer Lichtschranke unter der Magazinplatte erkannt wird. Damit die Farbbehälter genau positioniert werden können, ist diese Lichtschranke direkt hinter dem Vereinzelungsmechanismus angebracht. Eine weitere Lichtschranke erkennt die Nullposition des Magazins. Sie befindet sich aus Platzgründen auf der gegenüberliegenden Seite. Als Marker dienen kleine M2 Schrauben, die hinter dem Perlenbehälter in der Magazingrundplatte angebracht sind.

4.6. Steuerungselektronik

Die gesamte Elektronik der *BeadBox*, wie die Schrittmotoren und die Lichtschranken, werden von einem Arduino Leonardo Board kontrolliert. Um die einzelnen Komponenten auf einfache Weise zu verbinden, wurde eine gedruckte Schaltung entwickelt[2]. Die Schaltpläne und das Platinenlayout für die Schaltung befinden sich in Anhang B.

Die Schaltung besteht aus zwei Teilen. Der Hauptteil ist direkt mit dem Arduino verbunden. Der zweite kleinere Teil der Schaltung ist unterhalb des x-Achsenschlittens befestigt und wird über die bereits oben erwähnte Kabelschlaufe mit der Hauptschaltung verbunden. Die Elektronik muss mit einem 12 V Steckernetzteil mit mindestens 1,5 A betrieben werden. Die 12 V werden für die Schrittmotoren, den Getriebemotor und den Elektromagneten benötigt. Da die Ausgangssignale des Arduino allerdings nur eine Ausgangsspannung von 5 V besitzen, werden die Ausgangssignale des Arduino mithilfe von

[2]Die Schaltung entstand mit Unterstützung von Dr.-Ing. Frank Ziegler.

fertigen Schrittmotortreiberplatinen[3] auf 12 V verstärkt. Insgesamt sind drei Schrittmotortreiberplatinen in der *BeadBox* verbaut. Jede kann vier Leitungen betreiben. Zwei Schrittmotortreiberplatinen werden für die Ansteuerung der Schrittmotoren verwendet, die dritte ist für den Getriebemotor und den Elektromagneten zuständig. Weiterhin befindet sich auf der Hauptschaltung ein elektronischer Schaltregler, der die 12 V Betriebsspannung auf 5 V für die Lichtschranken herunter regelt. Ein USB-Anschluss ist zusätzlich integriert. Über ihn kann der Raspberry Pi mit Spannung versorgt werden. Dadurch ist kein weiteres Netzteil für den Raspberry Pi notwendig.

Ein wichtiger Aspekt der Schaltung ist die Ansteuerung der Lichtschranken, von denen sieben Stück in der *BeadBox* verbaut sind (vier für die xy-Positionierung, zwei für das Magazin und eine für die Vereinzelung). Es werden Gabellichtschranken vom Typ TCST4103 verwendet. Gabellichtschranken bestehen aus einer Leuchtdiode (Emitter, E), die ihr Licht auf einen Fototransistor (Detektor, D) sendet. Empfängt der Fototransistor Licht, wird er leitend. So kann man am Ausgang des Fototransistors testen, ob die Lichtschranke unterbrochen ist [13]. Die dazu benötigte Schaltung ist in Abbildung 9 dargestellt. Die Leuchtdiode des Emitters muss über einem Vorwiderstand an +5 V angeschlossen werden. Der Detektor-Anschluss wird mit einem Pull-Up-Widerstand mit +5 V verbunden. Der Pull-Up-Widerstand dient dazu, den Ausgang auf einen definierten Wert zu ziehen, wenn kein Licht auf den Fototransistor fällt [14].

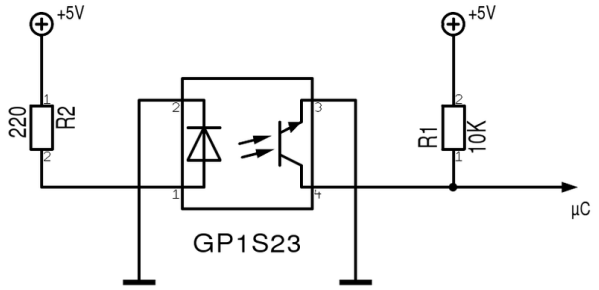


Abbildung 9: Beschaltung einer Gabellichtschranke, Quelle: www.rn-wissen.de

Für die Schaltungen wurde eine Platine entwickelt. Das Platinenlayout und die Schaltpläne wurden mit dem Programm *Eagle* der Firma CadSoft erstellt. Um das Platinenlayout auf die Platine zu bekommen, werden die Leiterbahnen zunächst auf ein Transparentpapier gedruckt. Dann wird eine Kupferfotoplatine mit der auf das Transparentpapier gedruckten Schaltung belichtet und anschließend entwickelt. Auf der Rückseite der Fotoplatine befindet sich eine lichtempfindliche Schicht, die durch das Belichten und Entwickeln an den nicht belichteten Stellen entfernt wird. Nach dem Entwickeln ist das Layout bereits auf der Kupferschicht zu erkennen. Anschließend wird die Platine in einem Ätzbad aus Eisen(III)-chlorid geätzt. Durch den Ätzvorgang wird das Kupfer

[3]Datenblatt: http://www.pollin.de/shop/downloads/D810027B.PDF

an den belichteten Stellen entfernt. Nun werden die Löcher für die Bauteile mit einem sehr feinen Bohrer (0,8 mm) gebohrt. Abschließend werden die Bauteile eingesetzt und verlötet [15].

Die fertige Hauptplatine besitzt auf der Unterseite Anschlüsse, die direkt in die Anschlüsse des Arduino gesteckt werden können. Für die Anschlüsse der Lichtschranken und Motoren, wurden 6-polige Wannenstecker mit Pfostenbuchsen und Flachbandkabel benutzt. Dies ermöglicht einen leichten Zusammenbau und den Austausch von Teilen. Für die Anbindung des x-Achsenschlittens wurden 16-polige Wannenstecker mit Pfostenbuchsen eingesetzt.

Die Platine für den x-Achsenschlitten wurde ebenfalls nach den oben beschriebenen Schritten erstellt. Sie ist allerdings sehr einfach gehalten. Auf ihr sind nur Anschlüsse vorhanden, da sich die wesentlichen Schaltungen auf der Hauptplatine befinden.

5. Software

Dieser Abschnitt behandelt den Aufbau der BeadBox-Steuerungssoftware. Zuerst wird die zugrundeliegende Architektur beschrieben. Anschließend werden die einzelnen Komponenten genauer erläutert.

5.1. Softwarearchitektur

Wie bereits im Entwurf der *BeadBox* erwähnt wurde, besteht die Software aus zwei wesentlichen Komponenten, der Hardwarekontrollsoftware, die auf dem Arduino läuft, und der Steuerungssoftware, die auf dem Raspberry Pi umgesetzt ist. Um dieser Tatsache gerecht zu werden, liegt der Software eine Schichtenarchitektur zugrunde (Abbildung 10). Diese sorgt zum einen für eine einfache Hardwareabstraktion und zum anderen für eine leichte Erweiterbarkeit.

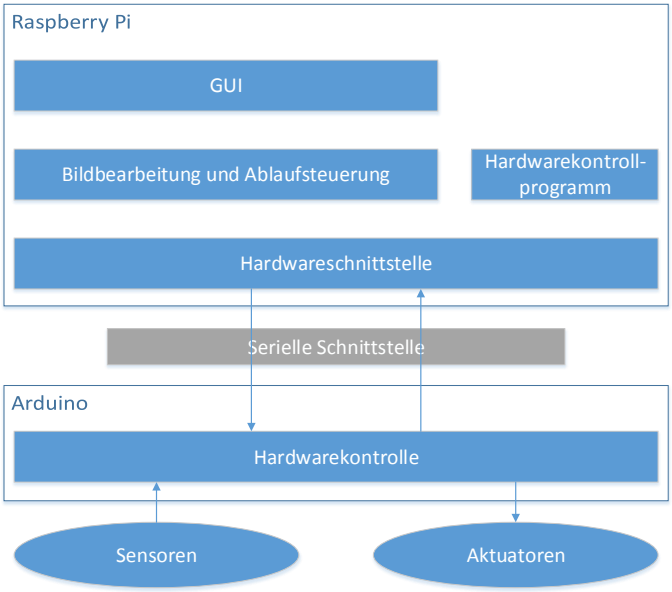


Abbildung 10: Softwarearchitektur

Die Software auf dem Arduino ist für die direkte Ansteuerung der Schritt- und Getriebemotoren sowie für die Abfrage der Lichtschranken zuständig. Auf dem Raspberry Pi läuft die eigentliche Anwendungssoftware, die wiederum in drei Schichten unterteilt ist. Die unterste Schicht kapselt die Kommunikation mit dem Arduino und stellt eine abstrahierte Schnittstelle zur Verwendung der Hardware zur Verfügung (Hardwareschnittstelle). Die Kommunikation mit dem Arduino wird mit einem selbstentwickelten Kommunikationsprotokoll über eine serielle Schnittstelle umgesetzt. Auf dieser Schicht setzt eine weitere Schicht auf, die für die Bildbearbeitung und die Ablaufsteuerung verantwortlich ist. Außerdem gibt es noch ein Hardwarekontrollprogramm, das nur die Hardwareschnittstellenschicht benutzt. Mit diesem Programm lässt sich die Hardware testen

und kalibrieren. Oberhalb der Bildbearbeitungs- und Ablaufsteuerungsschicht befindet sich die Benutzungsoberfläche, die eine einfache Bedienung der *BeadBox* ermöglicht.

Um eine möglichst gute Erweiterbarkeit innerhalb der einzelnen Schichten zu erreichen, sind die Schichten modular aufgebaut. Dazu wurde die Software nach dem *Dependency Injection Pattern* [18] entworfen. Klassen bekommen ihre Abhängigkeiten (engl.: dependendcy) im Konstruktor übergeben, anstatt sie selbst zu erzeugen. Dadurch wird die Software übersichtlicher, da Klassenabhängigkeiten direkt am Klassenkonstruktor ersichtlich sind. Zudem lassen sich so Implementierungen relativ leicht austauschen, da nur noch an einer Stelle Anpassungen vorgenommen werden müssen. Außerdem werden dort, wo spätere Erweiterungen besonders wahrscheinlich sind, Interfaces und abstrakte Klassen verwendet.

5.1.1. Projektstruktur

Bevor die Softwarekomponenten genauer beschrieben werden, ist es erst einmal wichtig zu wissen, wie der Quellcode im Projekt organisiert ist. Die Projektstruktur resultiert aus der gewählten Schichtenarchitektur und sieht wie folgt aus:

- **arduino** - In diesem Projekt (im Arduino-Jargon: Sketch) wird die Hardwarekontrollschicht implementiert. Wichtige Komponenten sind hier die Schrittmotorsteuerung, die Getriebemotorsteuerung (PWM), die Sensorabfrage und die serielle Kommunikation.

- **beadbox** - Dieses Projekt dient der einfachen Kompilierung der Anwendungssoftware und fasst die einzelnen Schichten zusammen.

  - **bbcore** - Dieses Projekt stellt die Funktionen der Hardwareschnittstellenschicht als Bibliothek zur Verfügung. Wichtige Aufgaben dieser Schicht ist die Kapselung der seriellen Kommunikation und die Bereitstellung von abstrahierten Hardwareschnittstellen.

  - **bbcontrol** - Dieses Programm bietet eine graphische Oberfläche zur direkten Steuerung der Hardware über die **bbcore**-Bibliothek. Es wird zum Testen und zur Kalibrierung der Hardware benötigt.

  - **bbprocessing** - Diese Bibliothek implementiert die Bildbearbeitungs- und Ablaufsteuerungsschicht. Sie setzt auf den Funktionen der **bbcore**-Bibliothek auf.

  - **bbgui** - In diesem Projekt wird die graphische Benutzungsoberfläche realisiert. Es dient in erster Linie der einfachen Benutzerführung und greift dazu auf die Schnittstellen der **bbprocessing**-Bibliothek zurück.

  - **bbconsole** - Dieses Projekt stellt eine einfache Konsolenanwendung zur Verfügung. Mit ihr kann ein Druckvorgang von anderen Programmen angestoßen werden. Es greift ebenso wie das **bbgui**-Projekt auf die Schnittstellen der **bbprocessing**-Bibliothek zurück.

In den folgenden Abschnitten werden nun die einzelnen Schichten und deren Schnittstellen genauer erläutert. Dabei werden die Schichten von unten nach oben abgearbeitet.

5.2. Hardwarekontrolle

Wie bereits oben erwähnt, ist die Hardwarekontrolle die unterste Schicht in der Software. Sie ist für die Ansteuerung der Schritt- und Getriebemotoren und die Abfrage der Sensoren zuständig. Die Software ist in C++ und dem im Arduino verwendeten C++-Dialekt geschrieben.

Arduino-Programme, die mit der Arduino-Entwicklungsumgebung erstellt werden, benötigen zwei wesentliche Funktionen. Ein minimales Programm ist in Listing 1 dargestellt.

Listing 1: Arduino-Programmrahmen, Quelle: www.arduino.cc

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Es besteht nur aus den zwei Funktionen `setup` und `loop`. Die `setup`-Funktion wird einmal vor dem Start des Programms oder nach einem Reset ausgeführt. Sie dient der Initialisierung. Dort wird beispielsweise die serielle Schnittstelle und die I/O-Pins konfiguriert. In die `loop`-Funktion gehört der Hauptcode. Sie wird während des Betriebs kontinuierlich aufgerufen und ist quasi der Rumpf einer Endlosschleife.

Die *BeadBox* kann sich in drei verschiedenen Zuständen befinden. Im RESET-Zustand wird die Positionierungseinheit und das Magazin in die Nullposition bewegt und alle internen Werte zurückgesetzt. Im POS-Modus wird das Magazin eingestellt und Stiftposition angefahren. Der letzte ist der IDLE-Zustand, in ihm wird lediglich auf neue Befehle gewartet. Im Allgemeinen werden in der Endlosschleife folgende Aufgaben erledigt:

1. Abfrage der seriellen Schnittstelle und Verarbeitung der Befehle
2. Zustandsabhängige Bearbeitung von Aktionen
   - IDLE: Pause von 20 ms
   - POS: Magazin- und xy-Positionierung und Überwachung
     Pause von 5 ms
   - RESET: Zurücksetzen in Nullposition
     Pause von 5 ms

Parallel zur Endlosschleife gibt es einen Timer-Interrupt, der alle 185 μs aufgerufen wird und zeitkritischen Aufgaben, wie die Taktung der Schrittmotoren und die Pulsweitenmodulation, übernimmt.

5.2.1. Sensorabfrage

Sensoren werden in der *BeadBox* zur Kalibrierung und zur Überwachung eingesetzt. Durch die Library des Arduino ist die Sensorabfrage sehr einfach [5]. Die Library stellt die Funktion `digitalRead(int pin)` zur Verfügung. Mit dieser Funktion lässt sich der digitale Wert, der an einem Eingangspin liegt, abfragen. Im Arduino Leonardo sind die Pins von 0 - 20 bzw. für die analogen Eingangspins von A0 - A5 durchnummeriert. Durch Übergabe der entsprechenden Pinnummer an `digitalRead` kann der aktuell anliegende digitale Wert abgefragt werden.

Listing 2: Sensorabfrage

```
#define TEST(pin) (digitalRead(pin) == HIGH)

...

// test sensor value
if (TEST(pin)) {
  ...
}
```

Die Lichtschranken in der *BeadBox* sind so angeschlossen, dass sie bei einer Unterbrechung `HIGH` (digital eins) sind. In Listing 2 ist das in der Software verwendete Makro zur Sensorabfrage zu sehen. Es wurde ein Makro anstatt einer Funktion verwendet, um den Overhead, der bei Funktionsaufrufen entsteht, in zeitkritischen Situationen zu vermeiden. Das Makro überprüft den Wert, der an einem Pin anliegt, und liefert `true`, wenn der Eingang `HIGH` (Lichtschranke unterbrochen) ist.

5.2.2. Pulsweitenmodulation

Die Drehzahl des Getriebemotors, der für das Farbmagazin verwendet wird, ist so hoch, dass die Marker des Farbmagazins nicht mehr von den Lichtschranken erkannt werden. Aus diesem Grund muss die Geschwindigkeit reduziert werden.

Pulsweitenmodulation (kurz PWM von engl. *pulse width modulation*) ist eine Technik, die häufig zur elektronischen Geschwindigkeitsregelung von Motoren verwendet wird. Bei der Pulsweitenmodulation werden Spannungspulse mit variabler Breite erzeugt (Abbildung 11). Dadurch kann die Geschwindigkeit eines Motors gesteuert werden, ohne Widerstände oder Spannungsregler verwenden zu müssen. Wichtige Kennwerte der Pulsweitenmodulation ist die Frequenz und das Tastverhältnis (engl.: *duty cycle*).

Je kürzer die Pulse sind desto langsamer bewegt sich der Motor, da er nur kurz eingeschaltet aber lange ausgeschaltet ist. Mit zunehmender Pulsweite nimmt auch die Geschwindigkeit des Motors zu [16].

Da Pulsweitenmodulation oft eingesetzt wird, besitzt der Arduino Leonardo 7 PWM-Pins (Pin 3, 5, 6, 9, 10, 11 und 13). Diese verwenden die internen Hardwaretimer des Microkontrollers, um die Pulsweite zu steuern. Da der Timer1 des Arduino jedoch für die Schrittmotorsteuerung verwendet wird und die Pins, die von anderen Timern gesteuert werden, bereits anderweitig belegt sind, muss eine eigene PWM-Implementation im Timer-Interrupt verwendet werden (Listing 3).

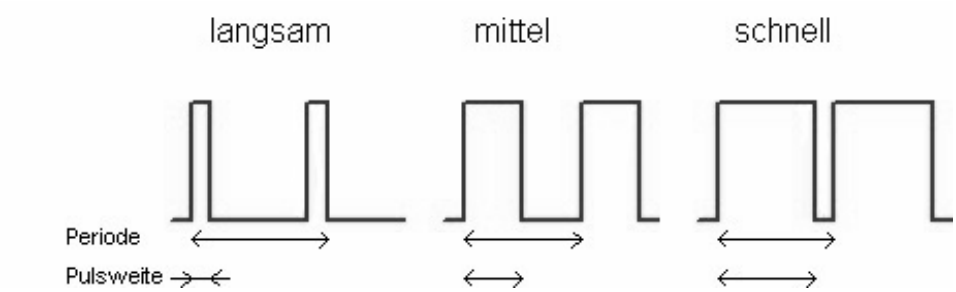


Abbildung 11: Pulsweitenmodulation, Quelle: http://www.rn-wissen.de/

Listing 3: Pulsweitenmodulation im Timer-Interrupt

```
void MagazinMotor::pwm(unsigned char value) {
  if (m_direction == 0)
    return; // nothing to do here

  if (value < m_speed) {
    // turn motor on
    if (m_direction == +1) {
      digitalWrite(m_pin1, HIGH);
      digitalWrite(m_pin2, LOW);
    } else {
      digitalWrite(m_pin1, LOW);
      digitalWrite(m_pin2, HIGH);
    }
  } else {
    // turn motor off
    digitalWrite(m_pin1, LOW);
    digitalWrite(m_pin2, LOW);
  }
}
```

Im Timer-Interrupt wird ein Zähler hoch gezählt, der der Funktion `pwm` des Magazinmotors übergeben wird. Anhand dieses Wertes werden die Pins angesteuert. Die PWM-Implementation kann den Magazingetriebemotor sowohl vorwärts als auch rückwärts bewegen. Dazu wird entweder der erste Motorpin (vorwärts) oder der zweite Motorpin (rückwärts) gesteuert. Die Implementation benutzt eine Frequenz von 5,4 kHz (Timer-Interrupt alle 185 μs). Es sind 10 verschiedene Tastverhältnisse möglich. Das Tastverhältnis sollte jedoch über 40 % liegen, d.h. ein Puls sollte über 740 μs lang sein, da die Spannung sonst zu niedrig für den Betrieb des Motors ist.

5.2.3. Schrittmotorsteuerung

Die Schrittmotorsteuerung wird ebenfalls im Timer-Interrupt erledigt. Allerdings wird ein Schritt nur in jedem zehnten Timer-Interrupt ausgeführt, da die Schritte mit einer höheren Frequenz nicht mehr sauber von den Schrittmotoren ausgeführt werden. Es wird also alle 1850 μs ein Schritt ausgeführt.

In Listing 4 ist ein Teil der Implementierung zu sehen. Solange noch ein Schritt übrig

ist, wird der Schrittzähler je nach gewünschter Drehrichtung des Motors inkrementiert bzw. dekrementiert. Um Überläufe des Schrittzählers zu vermeiden, wird er auf einen Bereich von 0 bis 20000 eingegrenzt. Die Funktion `doStep` setzt nun das in Tabelle 1 dargestellte Ansteuerungsschema um. Die jeweiligen Spulenschaltungen werden periodisch wiederholt, bis der Motor die gewünschte Position erreicht hat. Ist dies der Fall, werden alle Spulen in der `noStep`-Funktion abgeschaltet, damit die Motoren keinen Strom verbrauchen.

Listing 4: Schrittmotoransteuerung

```
if (m_stepsLeft > 0) {
  // modify step counter
  if ((m_direction == +1) && (++m_stepNumber > 20000))
    m_stepNumber = 0;

  if ((m_direction == -1) && (--m_stepNumber < 0))
    m_stepNumber = 20000;

  // test if all steps are done
  if (--m_stepsLeft == 0)
    m_direction = 0;

  // do one step
  doStep(m_stepNumber & 0x03);
} else {
  noStep();
}
```

Für die Steuerung der Schrittmotoren wurde eine eigene Klasse erstellt. Die Klasse hält die Schrittzähler und stellt die Funktionen für das Starten, Stoppen und Zurücksetzen bereit. Durch die Klasse wird auch erreicht, dass die zwei Schrittmotoren der Positionierungseinheit ohne weiteres gleichzeitig gesteuert werden können. Dadurch ist eine schnelle Positionierung möglich.

5.2.4. Magazinsteuerung

Das Magazin wird durch einen Getriebemotor gedreht. Die Drehgeschwindigkeit wird mithilfe der Pulsweitenmodulation reduziert. Zur Positionierung des Magazins wird eine Lichtschranke eingesetzt. Wenn sich ein Perlenbehälter über dem Vereinzelungsmechanismus befindet, wird sie unterbrochen. Eine weitere Lichtschranke erkennt die Nullposition des Magazins.

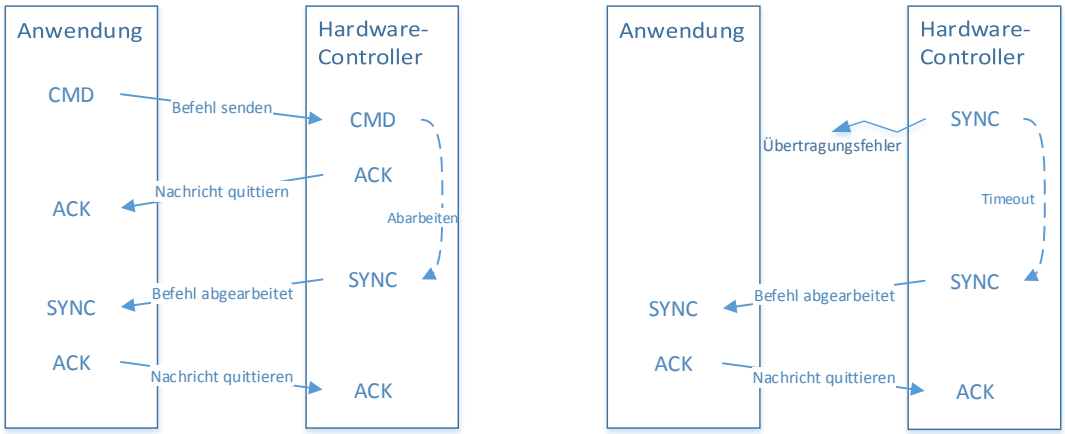
Um einen bestimmten Perlenbehälter auszuwählen, wird zuerst die Drehrichtung bestimmt. So hält man die Positionierungsdauer möglichst gering. Befindet sich der gewünschte Behälter vor dem aktuellen Behälter, muss sich das Magazin rückwärts drehen, andernfalls vorwärts. Anschließend wird die interne Magazinnummer bei jeder Lichtschrankenunterbrechung je nach Drehrichtung erhöht oder erniedrigt. Ist der gewünschte Farbbehälter erreicht, wird der Motor gestoppt und eine Feinpositionierung durchgeführt. Dabei wird der Motor kurz weiter gedreht und anschließend langsam in die

entgegengesetzte Richtung gedreht, bis die Lichtschranke wieder unterbrochen ist. Anschließend wird der Motornachlauf durch einen kalibrierten Wert ausgeglichen. So wird sichergestellt, dass die Öffnung des Behälters exakt über der Öffnung des Vereinzelungsmechanismusses liegt.

5.2.5. Kommunikation

Die Hardwarekontrollsoftware auf dem Arduino erhält die Befehle über eine serielle Schnittstelle. Dazu wird ein selbst entwickeltes Protokoll verwendet, welches die Synchronisation mit der Steuerungssoftware gewährleistet.

Eine Nachricht besteht aus einem Byte für den Befehl, einem Byte für die Nachrichten-Id und zwei mal zwei Byte für Nutzdaten. Um Befehlsbytes von den Bytes der Nachrichten-Id und der Nutzdaten unterscheiden zu können, werden die Befehlsbytes mit einer Eins im höchstwertigen Bit markiert. Die Nachrichten-Id und die Nutzdatenbytes dürfen dementsprechend keine Eins im höchstwertigen Bit haben. Damit dies gewährleistet ist, müssen die Nutzdaten vor dem Senden auf jeweils drei Byte aufgeteilt werden. Eine gesendete Nachricht besteht also insgesamt aus 8 Byte: 1 Befehlsbyte, 1 Byte für die Id, 3 Byte für das erste Nutzdatenpaket und 3 Byte für das zweite Nutzdatenpaket.



(a) Erfolgreiche Übertragung **(b)** Fehlerhafte Übertragung

Abbildung 12: Kommunikationsprotokoll

Jede Nachricht wird sobald sie empfangen wurde mit einer Acknowledge-Nachricht (`CMD_ACK`) quittiert. Trifft diese Acknowlege-Nachricht nicht ein, wird die Nachricht nach einem Timeout erneut gesendet. Dadurch wird sichergestellt, dass eine Nachricht auf jeden Fall ankommt. Um eine doppelte Abarbeitung von Befehlen zu vermeiden, wird im Arduino eine Liste mit den zuletzt eingetroffenen Nachrichten-Ids gepflegt. Ist eine Nachrichten-Id bereits vorhanden, wird der Befehl ignoriert und lediglich eine Acknowlege-Nachricht gesendet. Diese Liste wird bei jedem Verbindungsaufbau initialisiert. Nachdem ein Befehl vom Arduino abgearbeitet wurde, wird dies der Anwen-

dungssoftware mit einer Synchronisationsnachricht (`CMD_SYNC`) mitgeteilt. In der Anwendungssoftware kann so auf das Ende der Abarbeitung eines Befehls gewartet werden. Das erste Nutzdatenpaket einer Synchronisationsnachricht enthält die Befehls-Id, das zweite enthält befehlsspezifische Daten.

Listing 5: Befehlsdefinition und Beschreibung

```
...
// system commands
#define CMD_ACK   0x7F // data1: the received msg id ; data2: --
#define CMD_SYNC  0x7E // data1: the finished cmd id ; data2: command specific
#define CMD_RESET 0x7D // data1: --                  ; data2: --

#define CMD_INIT  0x7C // data1: --                  ; data2: --

// stepper motor commands
#define CMD_START_STEPPER1 0x00 // start stepper motor 1          - data1: step count
    ; data2: --  -> sync data2: current step count
#define CMD_STOP_STEPPER1  0x02 // stop stepper motor 1           - data1: --
    ; data2: --  -> sync data2: current step count
#define CMD_GET_STEPPER1   0x04 // get stepper motor 1 step count - data1: --
    ; data2: --  -> sync data2: current step count
...
```

Die möglichen Befehle sind in `arduino/serialcommands.h` definiert (Listing 5). Dort wird auch beschrieben, welche Funktion ein Befehl hat und welche Daten erwartet werden. Außerdem sind die in der Synchronisationsnachricht gesendeten Synchronisationsdaten beschrieben. Beispielsweise enthält die Synchronisationsnachricht für den Befehl `CMD_START_STEPPER1` zum Starten eines Schrittmotors den aktuellen Wert des Schrittzählers.

Die Nachrichten werden in der Hauptschleife des Arduino-Programmes abgefragt. Wurde ein Byte im einer Eins im höchstwertigen Bit empfangen, wird auf die folgenden sieben Byte gewartet. Sind auch diese empfangen, wird die Nachricht dekodiert. Anschließend wird der Befehl interpretiert und bearbeitet.

5.3. Hardwareschnittstelle

Die Hardwareschnittstellenschicht ist die unterste Schicht der eigentlichen Steuerungssoftware und setzt auf der Hardwarekontrollschicht auf. Sie dient der Kapselung der Kommunikation mit dem Arduino und stellt abstrahierte Schnittstellen zur Kontrolle der einzelnen Hardwarekomponenten bereit.

5.3.1. Kommunikationsmodul

Das wichtigste Modul dieser Schicht ist das Kommunikationsmodul, dass die Kommunikation mit dem Arduino über das in Abschnitt 5.2.5 beschriebene Nachrichtenprotokoll erlaubt. Für die Ansteuerung der seriellen Schnittstelle wird das Qt-Modul *QSerialPort* verwendet [19]. *QSerialPort* ist eine Bibliothek, die Qt-Anwendungen eine betriebssyste-

munabhängige Schnittstelle für serielle Schnittstellen zur Verfügung stellt. Die Bibliothek unterstützt Windows, Linux und Mac OS X.

Das Kommunikationsmodul besteht aus drei Klassen. Die Klasse `BbSerialDevice` verwaltet die serielle Schnittstelle und bearbeitet ein- und ausgehende Nachrichten. Die Klasse `BbSerialMessage` repräsentiert eine Nachricht und ist für die Kodierung und Dekodierung zuständig. `BbSerialRequest` kapselt eine Befehlsanforderung.

Ein wichtiges Feature des Kommunikationsmoduls ist die Möglichkeit, auf das Ende einer Befehlsabarbeitung zu warten. Dies dient dazu, das Drucksystem mit der Software zu synchronisieren, und ist notwendig, da die Kommunikation asynchron abläuft. So kann beispielsweise gewartet werden, bis ein Schrittmotor seine Schritte ausgeführt hat. Ein solches Szenario ist in Listing 6 zu sehen. Erst wird eine Anforderung (Request), den ersten Schrittmotor 1000 Schritte ausführen zu lassen, an den Arduino gesendet. Die Funktion `request` der Klasse `BbSerialDevice` sendet die Nachricht und liefet ein `BbSerialRequest`-Objekt zurück. Über dieses Objekt kann mit der Funktion `waitUntilSyncReceived` auf die Ankunft der Synchronisationsnachricht (`CMD_SYNC`) gewartet werden. Ein `BbSerialRequest` enthält neben der Antwort auch die ursprünglich gesendete Nachricht.

Listing 6: Absenden einer Nachricht und Warten auf Befehlsende

```
short steps = 1000;

// create message
BbSerialMessage startStepper1(CMD_START_STEPPER1,
    steps /* steps to move */,
    0     /* unused */));

// send message to arduino
BbSerialRequest request = arduino->request(startStepper1);

// do something here
...

// wait for the sync message
request.waitUntilSyncReceived();
```

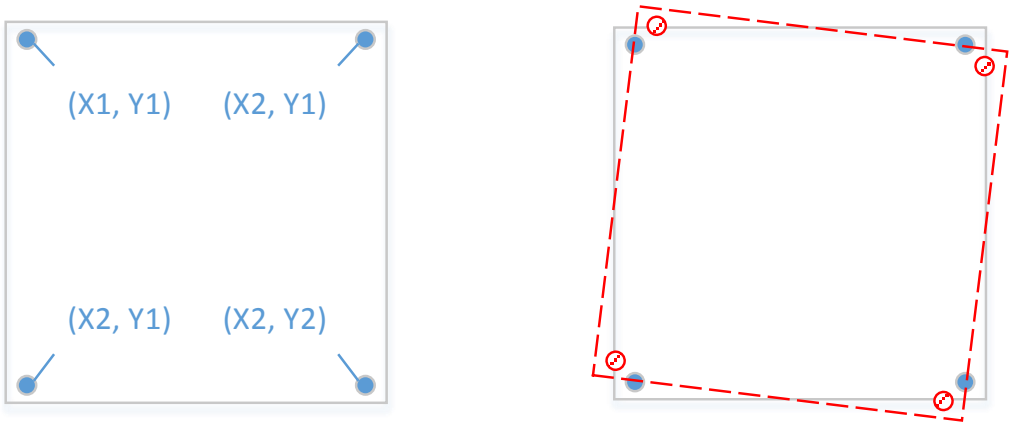
Realisiert wird der Wartemechanismus in der Klasse `BbSerialDevice` und in der Klasse `BbSerialRequest`. In der Klasse `BbSerialDevice` wird eine Liste mit den aktuell nicht beendeten Anforderungen gehalten. Von jedem Anforderungstyp (Befehl) kann jeweils nur einer zur Zeit aktiv sein. In der Funktion `waitUntilSyncReceived` wird mithilfe eines `QEventLoop`s auf das Eintreffen einer Synchronisationsnachricht gewartet. Eine Verwendung von Semaphoren oder ähnlichen Mechanismen ist nicht möglich, da sie das in Qt verwendete Signal und Slot-System blockieren. Wenn eine Synchronisationsnachricht empfangen wird, wird das zur Synchronisationsnachricht gehörende `BbSerialRequest`-Objekt in der Liste gesucht und aus der Liste entfernt. Anschließend wird dem Objekt die Antwort übergeben und die Warteschleife des `QEventLoop`s beendet.

Auf dem Kommunikationsmodul setzen die Klassen zur Kontrolle der Hardwarekomponenten auf. Sie sind Teil der Schnittstelle der Schicht. Es gibt drei verschiedene kon-

trollierbare Hardwarekomponenten: die Positionierungseinheit, den Vereinzelungsmechanismus und das Farbmagazin.

5.3.2. Kontrolle der Positionierungseinheit

Für die Positionierungseinheit sind zwei Kontrollklassen vorhanden, erstens die Klasse `BbPositioningCalibration` und zweitens die Klasse `BbPositioningControl`. Die erste Klasse bietet einen direkteren Zugriff auf die Positionierungseinheit. Sie kann zur Kalibrierung der Positionierungseinheit verwendet werden, da die Achsen schrittweise bewegt werden können. Außerdem kann der aktuelle Schrittzähler der Achsenschrittmotoren abgefragt werden. Mit der Klasse `BbPositioningControl` kann die Positionierungseinheit hingegen abstrahiert gesteuert werden. Sie bietet mit der `move`-Funktion eine Möglichkeit eine bestimmte Stiftposition auf der Stiftplatte anzufahren. Dazu werden die Start- und Endwerte der Schrittmotoren benötigt. Diese Werte müssen in einem manuellen Kalibrierungsvorgang ermittelt werden (Abbildung 13).



(a) Start- und Endwerte der Schrittmotoren **(b)** Verdrehung der y-Achse

Abbildung 13: Kalibrierung der Positionierungseinheit

Für die Kalibrierung wird das Drucksystem zunächst in die Nullposition gesetzt. Anschließend werden die Eckstifte der Stiftplatte angefahren und die jeweiligen Schrittzähler notiert. Durch Ungenauigkeiten in der Hardware ist die Führung der y-Achse leicht gegenüber der x-Achse verdreht (Abbildung 13b). Daraus ergibt sich eine leichte Verschiebung bei den äußeren Stiftpositionen. Diese Verschiebung wird ebenfalls gemessen und fließt in die Berechnung der Stiftpositionen ein. Aus den Start- und Endwerten, den Ausgleichsfaktoren und der Stiftplattengröße werden die absoluten Schrittmotorschritte ausgehend von der Nullposition berechnet (Listing 7). Aus diesen Werten werden die relativen Schrittwerte von der aktuellen Position der Schrittmotoren berechnet.

Listing 7: Berechnung der Schrittmotorschritte

```
double xPercent = ((double) x / (double) (m_width-1));
double yPercent = ((double) y / (double) (m_height-1));

// calculate absolute step count
int xStep = m_x1StepCount + qRound((double) (m_x2StepCount - m_x1StepCount) *
    xPercent);
int yStep = m_y1StepCount + qRound((double) (m_y2StepCount - m_y1StepCount) *
    yPercent);

// adjust step count
xStep += (m_xAdjust * yPercent);
yStep += (m_yAdjust * xPercent);
```

Anschließend führen beide Schrittmotoren die benötigten Schritte aus. Diese Aktion ist asynchron. Mithilfe der Funktion `waitUntilFinished` kann gewartet werden, bis die gewünschte Position erreicht ist. Zusätzlich bietet die Postionskontrolle die Möglichkeit, das Perlenplatzierungsrohr kurz hin und her zu bewegen. Das ist eine Maßnahme zur Fehlerbehebung. Damit kann eine Perle, die nicht korrekt auf den Stift gefallen ist, in die richtige Position bewegt werden.

5.3.3. Kontrolle des Vereinzelungssystems

Der Vereinzelungsmechanismus kann über die Klasse `BbSeperationControl` gesteuert werden. Diese Klasse stellt zwei Funktionen zur Verfügung. Mit der Funktion `next` wird eine Perle vereinzelt. Dieser Vorgang ist fehleranfällig und muss deswegen überprüft werden. Der Befehl für den Arduino sieht deswegen eine Fehlerrückmeldung vor (Listing 8).

Listing 8: Vereinzelungsbefehle und Fehlercodes

```
...
// seperation commands
#define CMD_NEXT_BEAD  0x06 // seperate bead                    - data1: --  ; data2:
    --  -> sync data2: the seperation error code (defined below)
#define CMD_CHECK_BEAD 0x07 // check if bead is still in tube - data1: --  ; data2:
    --  -> sync data2: the seperation error code (defined below)

// seperation error codes
#define E_SUCCESS  0 // bead is places successful
#define E_WOBBLE   1 // bead is not placed properly
#define E_EMPTY    2 // the magazin is empty
...
```

Es gibt drei Fehlercodes. Der erste ist `E_SUCCESS`. Dieser Code wird zurückgeliefert, wenn kein Fehler aufgetreten ist. Wurde die Perle aber nicht korrekt abgesetzt und befindet sich noch im Perlenplatzierungsrohr, wird der Fehlercode `E_WOBBLE` zurückgeliefert. Dieser Code teilt der Software mit, dass das Perlenplatzierungsrohr hin und her bewegt werden soll, um die Perle auf den Stift fallen zu lassen. Der letzte Fehlercode `E_EMPTY` tritt auf, wenn keine Perle vereinzelt werden konnte. Dies kann passieren, wenn das Magazin leer ist. Da auch eine im Magazin verklemmte Perle die Ursache für diesen Fehler

sein kann, wird noch vier weitere Male versucht, eine Perle zu vereinzeln. Tritt der Fehler nach dem fünften Versuch immer noch auf, wird er als nicht behebbar eingestuft und eine höhere Schicht mit dem Problem betraut.

Mit der Funktion `check` der Vereinzelungskontrolle kann überprüft werden, ob sich noch eine Perle im Platzierungsrohr befindet. Diese Funktion dient der Fehlerbehebung, falls eine Perle nicht sauber auf den Stift gefallen ist. Sollte dies der Fall sein, wird das Perlenplatzierungsrohr durch die Positionierungseinheit sooft hin und her bewegt, bis die Perle korrekt auf dem Stift platziert werden konnte.

5.3.4. Kontrolle des Farbmagazins

Als letzte Hardwarekomponente fehlt noch das Farbmagazin. Die Magazinkontrolle wird in der Klasse `BbMagazinControl` implementiert. Mit der Funktion `setColor` kann ein Behälter über die Magazinnummer ausgewählt werden. Die Funktion macht nichts Weiteres, als den Befehl `CMD_SET_COLOR` an den Arduino zu senden. Der Auswahlvorgang wird asynchron ausgeführt. Mit der Funktion `waitUntilFinished` kann auf ihren Abschluss gewartet werden.

5.3.5. Supportklassen

Neben den Hardwarekontrollklassen bietet diese Schicht die Möglichkeit, auf die Konfigurationsdatei zuzugreifen. Diese Datei basiert auf dem von der Klasse `QSettings` implementierten Ini-Dateiformat. Dadurch ist sie plattformunabhängig und zudem leicht editierbar.

Der zentrale Einstiegspunkt für diese Schicht ist die Klasse `BbCore`. In der Funktion `initialize` wird die serielle Schnittstelle initialisiert, die Positionskontrolle mit den Konfigurationsdaten aus der Konfigurationsdatei kalibriert und die anderen Kontrollklassen instantiiert. Die Funktion `reset` führt einen Reset der *BeadBox* aus, d.h. die Positionierungseinheit und das Farbmagazin werden in die Nullposition gefahren und alle internen Zustände zurückgesetzt. Mithilfe der Funktion `standby` kann das Drucksystem in eine für den Transport geeignete Position gefahren werden.

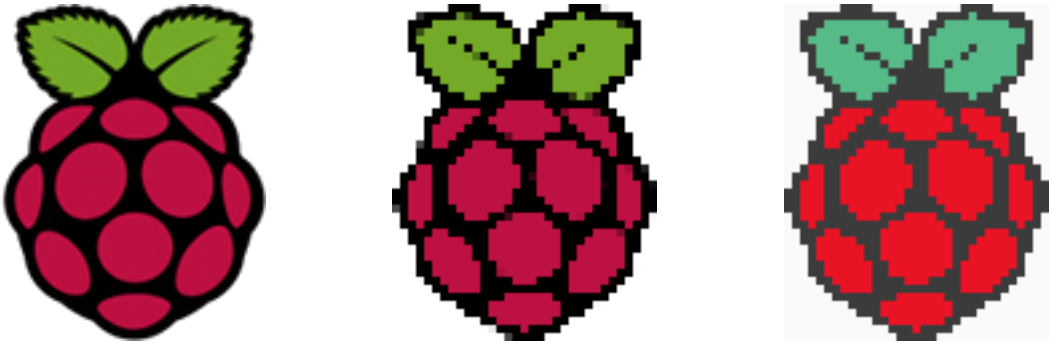
5.4. Bildverarbeitung und Ablaufsteuerung

Oberhalb der Hardwareschnittstellenschicht setzt die Bildverarbeitungs- und Ablaufsteuerungsschicht auf. Sie ist für die Anpassung des Ausgangsbildes an die Möglichkeiten des Drucksystems und die Steuerung des Druckablaufes verantwortlich.

5.4.1. Bildverarbeitung

Die Größe der Bügelperlenbilder, die die *BeadBox* erstellen kann, ist ebenso wie die Anzahl der Bügelperlenfarben begrenzt. Die Bilder müssen also vor dem Druckvorgang angepasst werden. Diese Anpassung läuft in der Regel in mehreren Stufen ab. Zuerst wird die Größe des Ausgangsbildes (Abbildung 14a) angepasst. Dazu wird die Funktion `scaled` der Klasse `QImage` aus dem Qt-Framework benutzt. Sie führt eine Skalierung

mit bilinearer Filterung durch. Diese Skalierungsmethode wurde gewählt, da sie zum einen einfach verwendet werden kann und zum anderen sehr gute Resultate liefert. Das Ergebnis dieses Vorganges ist in Abbildung 14b dargestellt. Danach wird das Bild mit Bildfiltern für den Druckvorgang vorbereitet.



(a) Ausgangsbild **(b)** Anpassung der Größe **(c)** Anpassung der Farben

Abbildung 14: Anpassung des Ausgangsbildes

Ein Bildfilter muss die Funktion `getFilteredImage` des Interfaces `IImageFilter` implementieren. Dieser Funktion wird das zu filternde Bild und die Farbpalette übergeben. Das Ergebnis der Filterung wird zurückgeliefert.

Aktuell sind zwei Filter produktiv einsetzbar, ein einfacher Farbanpassungsfilter und ein Filter, der den Floyd-Steinberg-Dithering-Algorithmus implementiert. Der Farbanpassungsfilter ersetzt jeden Pixel des skalierten Ausgangsbildes mit der Farbe aus der Farbpalette, die den kleinsten euklidischen Abstand zur Farbe des Pixels hat. Der Floyd-Steinberg-Dithering-Algorithmus, der von dem anderen Filter implementiert wird, ist ein Dithering-Algorithmus, der 1976 von Robert W. Floyd und Louis Steinberg veröffentlicht wurde [24]. Dithering ist eine Technik, die durch Fehlerdiffusion versucht, eine Illusion der ursprünglichen Farbe zu erzeugen. Da diese Technik bei den kleinen Bildern eher schlechte Ergebnisse erzielt, ist der einfache Farbanpassungsfilter oft vorzuziehen.

Während der Entwicklung wurde mit weiteren Filtern experimentiert. Um den Bildern stärkere Konturen zu geben, wurde u.a. ein Schärfefilter und ein Kantenerkennungsalgorithmus ausprobiert. Beide Filter verbesserten das Ergebnis jedoch nicht. Oft entstanden sogar schlechter erkennbare Bilder.

5.4.2. Ablaufsteuerung

In der Ablaufsteuerung werden die Verfahren für den Druckvorgang implementiert. Dazu werden die Module der Hardwareschnittstellenschicht verwendet.

Die Klasse `BeadBoxPlotter` regelt den Druckvorgang. Sie benötigt den Bildfilter, der die Bildanpassung vornimmt, sowie die Palette mit den von der *BeadBox* unterstützten Farben. Die Funktion `prepareImage` liefert eine Vorschau des angepassten Bildes. Mit der Funktion `start` kann der Druckvorgang gestartet werden. Ein Druckvorgang hat in der Regel folgenden Ablauf:

1. Anwendung der Bildfilter auf das Bild
2. Zurücksetzten der Hardware in die Nullposition.
3. Druckvorgang durchführen
4. Zurücksetzten der Hardware in die Nullposition.

Zuerst wird das Ausgangsbild mithilfe der Bildfilter vorbereitet. Anschließend wird die Maschine zurückgesetzt, da der aktuelle Zustand der Maschine nicht klar ist. Nachdem der Druckvorgang abgeschlossen ist, wird die Maschine erneut zurückgesetzt, um das Entnehmen der Stiftplatte zum Bügeln zu erleichtern. Für die Durchführung des Druckvorgangs kann man zwischen zwei verschiedenen Verfahren wählen. Es gibt die Möglichkeit, beliebige weitere Verfahren zu implementieren.

Die Klasse, die ein Druckverfahren implementiert, muss von der abstrakten Klasse `AbstractPlotterMethod` abgeleitet sein. Diese Klasse stellt grundlegende Funktionen zum Abbrechen, Pausieren und Fortfahren des Druckvorgangs zur Verfügung. Die Funktion `plot` muss von der Subklasse überschrieben werden (Listing 9). In ihr wird das Druckverfahren implementiert.

Listing 9: Deklaration der plot-Funktion

```
virtual PlotterStatistics plot(
    const QImage &image,
    BbMagazinControl *magazinControl,
    BbPositioningControl *positioningControl,
    BbSeperationControl *seperationControl,
    ColorPalette *colorPalette) = 0;
```

Die `plot`-Funktion bekommt das vorbereitete Bild, die Hardwarekontrollklassen und die Farbpalette übergeben. Als Rückgabewert liefert die Funktion ein Objekt der Klasse `PlotterStatistics`. Mit dieser Klasse können während des Druckvorgangs Statistiken, wie Gesamtdauer, Anzahl der abgesetzten Perlen, etc., gesammelt werden. Bei der Implementation der `plot`-Funktion muss man einige Bedingungen einhalten, damit es zu keinen Problemen kommt. Mithilfe der Signale `sigUpdatePosition` und `sigUpdateProgress` sollte die aktuelle Position und der Fortschritt des Druckvorgangs mitgeteilt werden. Zudem muss im Falle eines Fehlers während der Vereinzelung das Signal `sigSeperationError` gesendet werden, damit der Benutzer benachrichtigt wird. Der Algorithmus muss pausier- und abbrechbar sein. Der Zustand kann über die Funktion `state` der Klasse `AbstractPlotterMethod` überprüft werden. Wurde der Algorithmus pausiert, befindet er sich im Zustand `Paused`. In diesem Fall muss mit der Funktion `wait` auf das Fortsetzen gewartet werden. Ist der Zustand `Aborted` gesetzt, sollte der Druckvorgang unmittelbar abgebrochen werden. Auch im Falle eines Vereinzelungsfehlers muss gewartet werden. Ist das Flag `skip` gesetzt, sollte der Setzvorgang der aktuellen Perle übersprungen werden. Eine beispielhafte Implementation dieser Bedingungen ist in Listing 10 zu sehen.

Listing 10: Beispielhafte Implementierung der plot-Funktion

```
while (!ready) {
  // handle states
  if (state() == AbstractPlotterMethod::Aborted)
    return; // plotting is aborted
  else if (state() == AbstractPlotterMethod::Paused)
    wait(); // plotting is paused

  if (!skip()) {
    // plotting steps here.
  }

  // update position
  emit sigUpdatePosition(x, y);

  // update progress
  emit sigUpdateProgress((++current) / total);
}
```

Die *BeadBox* unterstützt zur Zeit einen zeilenweisen und einen farbweisen Druck. Der zeilenweise Druck ist relativ trivial. Bei diesem Verfahren wird das Bild wie bei einem Drucker zeilenweise von oben nach unten gedruckt. Dazu muss natürlich immer die richtige Farbe ausgewählt werden. Da dies bei sehr farbhaltigen Bildern jedoch zu einem ineffizienten Hin- und Herwechsel der Farben führt, wurde ein farbweises Druckverfahren implementiert. Dieses Verfahren druckt die Farben nacheinander. Dazu wird das Bild vorverarbeitet und alle Pixel einer Farbe in einer Liste gespeichert. Die Farben sind nach ihrer Position im Farbmagazin sortiert, sodass der zeitraubende Farbwechsel auf ein Minimum reduziert wird. Bei der Positionierung wird nach der Best-First-Methode vorgegangen, d.h. es wird immer der Stift mit der aktuell geringsten Entfernung als nächstes angesteuert. Dies führt zwar nicht zu einem optimalen Weg, reduziert aber den Positionierungsaufwand erheblich.

5.5. Benutzeroberfläche

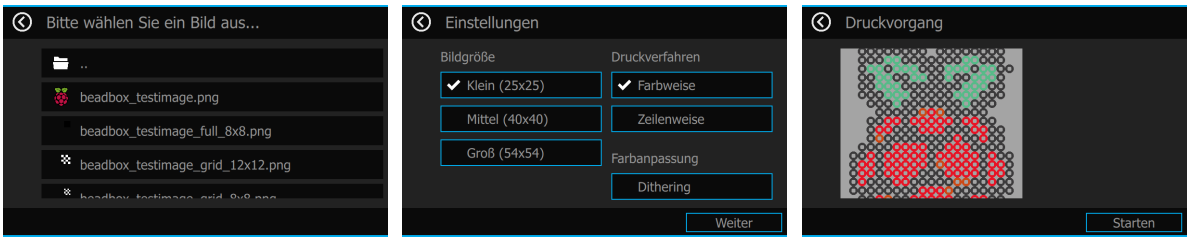
Über die Benutzeroberfläche der *BeadBox* lässt sich der Druckvorgang komfortabel und übersichtlich kontrollieren. Sie stellt die oberste Schicht in der Schichtenarchitektur dar und setzt auf den Funktionen der Bildverarbeitungs- und Ablaufsteuerungsschicht auf.

Da die Oberfläche für Touchscreens geeignet sein soll, ist sie in QML (Qt Modeling Language) und QtQuick umgesetzt. QML ist eine vom Qt-Projekt entwickelte deklarative Programmiersprache für graphische Benutzungsoberflächen und wurde in Qt 4.7 eingeführt. QML ist auf die Anforderungen der Programmierung von Mobilanwendungen zugeschnitten und unterstützt deswegen Toucheingaben und Animationen. Außerdem ist JavaScript in QML integriert, sodass vollständige Anwendungen in JavaScript und QML umgesetzt werden können. QML arbeitet aber auch über das Qt Meta Object System nahtlos mit C++-Code zusammen. QtQuick ist ein Ui-Framework und setzt auf QML auf. Es bietet grundlegende Gui-Komponenten, wie Rechtecke, Texte, Bilder, usw.. Mittlerweile ist QtQuick 2 (Qt 5) erschienen. Es bietet zahlreiche Verbesserungen gegenüber QtQuick 1, wie beispielsweise einen OpenGL-Scenegraph, der für eine flüssige Darstel-

lung der Oberfläche sorgt.

Die Oberfläche ist übersichtlich gestaltet und es werden durchgehend große Schaltflächen verwendet, um die Bedienung per Finger zu vereinfachen. Alle Komponenten werden relativ zueinander positioniert, wodurch die Größe der Benutzeroberfläche variierbar ist. Für die GUI wurde ein dunkles Farbschema gewählt, um eine Ähnlichkeit mit dem Design der Maschine (Aluminium und schwarzer Kunststoff) zu erreichen.

Die Oberfläche ist nach dem MVC-Prinzip aufgebaut. Die View ist komplett in QML und QtQuick umgesetzt, der Controller und die Models in C++. Die Models enthalten u.a. das Farbschema und die Einstellungen für den Druckvorgang. Ein Druckvorgang kann im Hauptmenü gestartet werden. Im Moment ist es möglich, entweder ein Bild als Vorlage oder die interne Zeichenfläche zu verwenden. Über die interne Zeichenfläche kann auf einfache Weise schnell ein Bild gezeichnet und anschließend gedruckt werden.



(a) Bildauswahl **(b)** Bildauswahl **(c)** Druckfortschritt

Abbildung 15: Starten eines Druckvorgangs von einer Bildvorlage

Ein Druck auf das entsprechende Symbol im Hauptmenü öffnet einen Wizard, der den Benutzer durch die notwendigen Schritte des Druckvorgangs führt (Abbildung 15). Um den Druckfortschritt zu visualisieren, wird ein aus der Bildvorlage erstelltes virtuelles Bügelperlenbildes angezeigt. Diese Vorschau wird während des Druckvorgangs kontinuierlich aktualisiert. Ist eine Perle erfolgreich gesetzt worden, wird die entsprechende Perle in der Vorschau hervorgehoben. Diese Fortschrittsanzeige vermittelt einen guten Eindruck des Druckvorgangs.

Tritt während des Druckvorgangs ein Fehler auf, erscheint eine Fehlermeldung, die den Benutzer auf den Fehler hinweist und einen Lösungsvorschlag anbietet. Über diese Fehlermeldung kann man den Druckvorgang auch abbrechen, falls der Fehler schwerwiegender sein sollte. Näheres zur Bedienung der *BeadBox* befindet sich im Benutzerhandbuch im Abschnitt 6.2.3.

5.6. Konsolenanwendung

Die Konsolenanwendung ist gegen Ende der Entwicklungsarbeiten entstanden. Sie setzt ebenso wie die Benutzeroberfläche auf der Bildverarbeitungs- und Ablaufsteuerungsschicht auf. Mit der Konsolenanwendung kann ein Druckvorgang auch von Drittanwendungen auf einfache Weise angestoßen werden. Die Konsolenanwendung hat folgende Paramenter:

- **-s, –size** Mit diesem Parameter kann die Größe des Bügelperlenbildes ausgewählt werden. Mögliche Werte sind „small“ (25x25 Perlen), „medium“ (40x40 Perlen) und „large“(54x54 Perlen). Der Standardwert dieses Parameters ist „small“.

- **-p, –plotter** Über diesen Parameter kann das Druckverfahren ausgewählt werden. Es gibt zwei alternative Werte: „line“, wenn der Druck zeilenweise ausgeführt werden soll und „color“, wenn der Druck farbweise ausgeführt werden soll. Letzterer Wert ist der Standard.

- **-d, –dither** Mithilfe dieses Parameters wird Dithering zur Farbanpassung verwendet.

- **–show-statistics** Das Programm zeigt nach erfolgreichem Druckvorgang Druckstatistiken an.

Als letzten Parameter benötigt die Anwendung das zu druckende Bild. Ein beispielhafter Druckvorgang ist in Listing 11 zu sehen. Dieser Aufruf druckt ein kleines Bild des Ausgangsbildes „testimage.png“ im farbweisen Druckverfahren und zeigt anschließend die Druckstatistiken an.

Listing 11: Druckvorgang mit der Konsolenanwendung

```
$ ./bbconsole --size small --plotter color --show-statistics ./testimage.png
```

Während des Druckvorgangs wird der Druckfortschritt sowie eventuelle Fehlermeldungen angezeigt.

5.7. Hardwarekontrollprogramm

Das Testprogramm entstand während der Entwicklung der Maschine. Es setzt auf der Hardwareschnittstellenschicht auf und ist in erster Linie dafür gedacht, die einzelnen Funktionen der Hardware gezielt testen zu können.

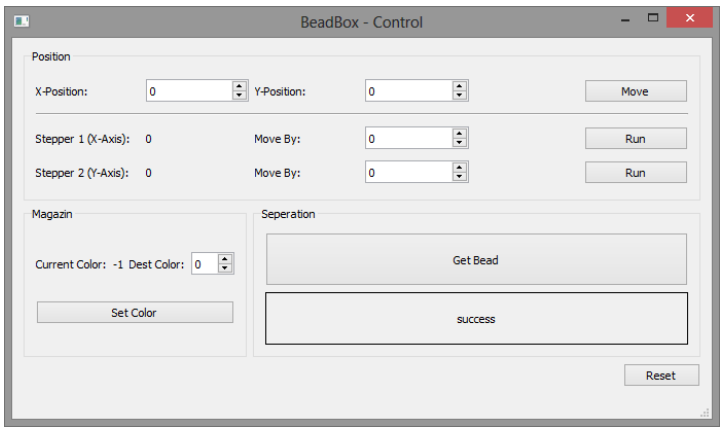


Abbildung 16: Hardwarekontrollprogramm

Es bietet Schaltflächen zur Steuerung der Positionierungseinheit, des Farbmagazins und des Vereinzelungssmechanismus an. Außerdem existieren Anzeigen für die internen Werte der Schittmotoren und der Farbmagazinnummer. Dieses Programm ist auch für die initiale Kalibrierung der *BeadBox* gedacht, da es dem Benutzer direkten Zugriff auf die Schrittmotoren gibt.

6. Einrichtung und Bedienung

In diesem Abschnitt wird das Aufsetzen der Entwicklungsumgebung sowie die Einrichtung und die Bedienung der *BeadBox* beschrieben.

6.1. Entwicklungsumgebung

Im Folgenden wird die Installation und Konfiguration des Raspberry Pi, die Erstellung des QtonPi-Images und der Cross-Kompilierungsvorgang der Anwendungssoftware erklärt. Als Betriebssystem für den Cross-Kompilierungsvorgang dient ein aktuelles Ubuntu 13.04 in der 64-bit Version.

6.1.1. Raspbian

Die Installation von Raspbian ist sehr einfach. Zunächst muss man das Raspbian-Image herunterladen (`http://www.raspberrypi.org/downloads`) und es danach auf eine SD-Karte mit mindestens 2 GB Speicherplatz schreiben. Unter Windows benötigt man dazu beispielsweise das Programm *Win32DiskImager* (`http://sourceforge.net/projects/win32diskimager`). Unter Linux kann dies mit dem Befehl *dd* erledigt werden.

```
$ sudo dd if=<raspian-image> of=<sd-card> bs=1M
```

Anschließend kann die SD-Karte in den Raspberry Pi gesteckt werden. Der Raspberry Pi ist nun einsatzbereit [22]. Für die Verwendung von Qt 5 auf dem Raspberry Pi muss man jedoch ein selbst erstelltes Image verwenden.

6.1.2. QtonPi

QtonPi ist eine an den Raspberry Pi angepasste Version von Qt 5. Sie bietet alle Features von Qt 5 inklusive OpenGL-Unterstützung für QtQuick und macht es so möglich, schnelle, dynamische und animierte Benutzungsoberflächen für den Raspberry Pi zu erstellen.

Die Binaries für *QtonPi* müssen selbst kompiliert werden. Dies kann entweder auf dem Raspberry Pi selbst erfolgen (*Native Build of Qt 5 on a Raspberry Pi* [10]) oder via Cross-Kompilation auf einem anderen Computer. Aufgrund des Umfanges des Qt-Projektes und der beschränkten Rechenleistung des Raspberry Pi und der daraus resultierenden Dauer des Kompiliervorgangs, ist die Cross-Kompilierung die bessere Alternative.

Eine Schritt-für-Schritt Anleitung zur Cross-Kompilierung ist der *Beginner's guide to cross-compile Qt5 on RaspberryPi* [11]. Zur vereinfachten Installation kann das `bakeqtpi`-Script verwendet werden, welches aus einem Git-Repository geklont werden kann.

```
$ git clone git://gitorious.org/bakeqtpi/bakeqtpi.git
```

Dieses Script übernimmt alle wesentlichen Schritte. Damit die neusten Softwarepakete Verwendung finden, sollte das Script noch angepasst werden:

```
28 #Raspbian image and download stuff
29 RASPBIAN=2013-07-26-wheezy-raspbian
```

```
...
54 QT5_PACKAGE_MAJ=5.1
55 QT5_PACKAGE_VER=5.1.1
56 QT5_PACKAGE_DOWNLOAD=http://download.qt-project.org/official_releases/qt/
       $QT5_PACKAGE_MAJ/$QT5_PACKAGE_VER/single/qt-everywhere-opensource-src-
       $QT5_PACKAGE_VER.tar.gz
```

Außerdem wird noch das QtSerialPort Modul aus dem Qt-Projekt benötigt. Da es nicht standardmäßig vom `bakeqtpi`-Script kompiliert wird, muss es in die Liste der zu kompilierenden Module mit aufgenommen werden:

```
484 QT_COMPILE_LIST="qtimageformats␣qtsvg␣qtjsbackend␣qtscript␣qtxmlpatterns␣
       qtdeclarative␣qtgraphicaleffects␣qtmultimedia␣qtquick1␣qtserialport"
```

Unter Ubuntu müssen noch einige Pakete installiert werden:

```
$ sudo apt-get install build-essential git ia32-libs
```

Anschließend wird der Kompiliervorgang gestartet:

```
$ ./bakeqtpi.bash --httppi --package
```

Das Script führt nun folgende Aktionen aus:

1. Das Raspbian-Image wird via HTTP geladen und entpackt (–httppi)
2. Das Raspbian-Image wird gemountet
3. Der Cross-Compiler wird wird heruntergeladen
4. Die Qt-Quelldateien werden als tar.gz-Archiv heruntergeladen und entpackt (–package)
5. Die Qt 5 Module werden kompiliert
6. Die Qt 5 Binaries werden auf in das Image kopiert

Nach dem erfolgreichen Kompiliervorgang muss das QtonPi-Raspbian-Image, wie im vorherigen Abschnitt beschrieben auf eine SD-Karte geschrieben werden.

6.1.3. Kompilierungsvorgang

Arduino-Software Für die Programmierung des Arduino wird die Arduino Entwicklungsumgebung (`http://arduino.cc/en/Main/Software`) benötigt. Mit ihr wird das Programm kompiliert und anschließend über eine serielle Schnittstelle auf das Board geladen. Dazu muss das richtige Board (bei der *BeadBox* das Arduino Leonard) unter `Tools > Board` und die serielle Schnittstelle unter `Tools > Serielle Schnittstelle` ausgewählt werden. Mit `Datei > Upload` wird das Programm kompiliert und auf das Board geladen.

BeadBox-Software Zunächst muss das aktuelle Installationspaket für Qt 5 heruntergeladen werden (http://qt-project.org/downloads). Die Software verwendet die Qt-Version 5.1.1. Anschließend wird der Installer ausführbar gemacht und gestartet:

```
$ sudo chmod +x ./qt-linux-opensource-1.4.0-x86-64-online.run
$ sudo ./qt-linux-opensource-1.4.0-x86-64-online.run
```

Nachdem die Installation abgeschlossen ist, muss der QtCreator für die Cross-Kompilierung konfiguriert werden. Der QtCreator ist eine im Qt Paket mitgelieferte IDE und befindet sich im Installationsverzeichnis unter `Tools/QtCreator/bin`. Zunächst muss im QtCreator der Cross-Compiler eingestellt werden. Dazu öffnet man den Einstellungsdialog über `Extras > Einstellungen`. Im Einstellungsdialog wählt man nun unter *Erstellung und Ausführung* den *Compiler*-Tab. Dort wird ein neuer GCC-Compiler eingerichtet:

Name:	ARM GCC
Compiler-Pfad:	/home/<user>/opt/gcc-4.7-linaro-rpi-gnueabihf/bin/arm-linux-gnueabihf-g++

Anschließend muss die Qt Version im *Qt Versionen*-Tab eingerichtet werden. Als Pfad wird „/usr/local/qt5pi/bin/qmake“ gewählt. Danach fügt man den Raspberry Pi als Gerät unter *Geräte* hinzu. Dazu muss ein neues *Generisches Linus-Gerät* hinzugefügt werde. Im Einrichtungswizard werden jetzt die entsprechenden Einstellungen für Hostname/IP-Adresse, Nutzername und Password angegeben. Abschließend wird unter *Erstellung und Ausführung* im *Kits*-Tab die Umgebung hinzugefügt:

Name:	Raspberry Pi
Gerätetyp:	Generisches Linux-Gerät
Geräte:	Raspberry Pi
Compiler:	ARM GCC
Debugger:	`Bearbeiten > Auswählen` /home/<user>/opt/gcc-4.7-linaro-rpi-gnueabihf/bin/arm-linux-gnueabihf-gdb
Qt-Version:	Qt 5.1.1 (qt5pi)

Jetzt ist es möglich, das BeadBox-Projekt zu öffnen und für den Raspberry Pi zu kompilieren. Dazu muss das QtonPi-Image auf „/home/<user>/opt/rasp-pi-rootfs“ gemounted werden. Ist der Raspberry Pi angeschlossen, wird das Programm automatisch auf den Pi kopiert.

6.2. Benutzerhandbuch

Im Folgenden wird die Inbetriebnahme der *BeadBox* und die Bedienung der Anwendungssoftware beschrieben.

6.2.1. Inbetriebnahme

Die *BeadBox* besteht aus dem Drucksystem (Druckvorrichtung und Arduino), dem Raspberry Pi und einem Touchscreen. Das Drucksystem darf beim Zusammenbau aus Sicherheitsgründen nicht eingesteckt sein.

Als erstes wird der Raspberry Pi mit dem Drucksystem verbunden. Dazu wird der Stromanschluss des Raspberry Pi mit einem USB-Kabel an den USB-Ausgang des Drucksystems angeschlossen. Dann wird der im Drucksystem verbaute Arduino an den unteren USB-Anschlus des Raspberry Pi angeschlossen. Danach wird der Touchscreen via HDMI und via USB mit dem Raspberry Pi verbunden. Abschließend darf das Drucksystem und der Touchscreen eingesteckt werden. Das System ist nun betriebsbereit.

6.2.2. Konfigurationsvorgang

Nach einer Änderung an der Hardware des Drucksystems, muss es neu kalibriert werden, um den Druckvorgang korrekt ausführen zu können. Im regulären Betrieb kann dieser Schritt übersprungen werden. Für die Kalibrierung wird das Programm `bbcontrol` (Abschnitt 5.7), welches sich im Programmordner der *BeadBox* befindet, benötigt. Die Kalibrierungswerte müssen manuell in die Konfigurationsdatei („settings.ini“) eingetragen werden.

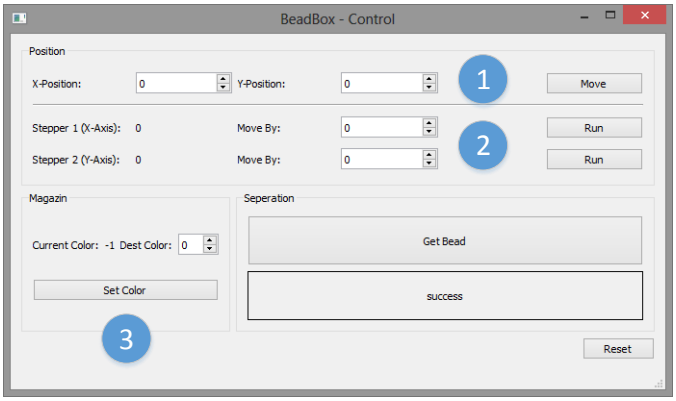
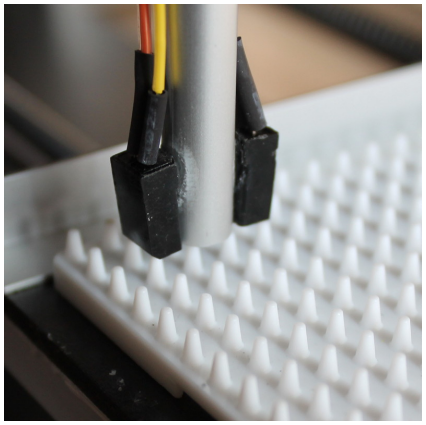


Abbildung 17: Hardwarekontrollprogramm

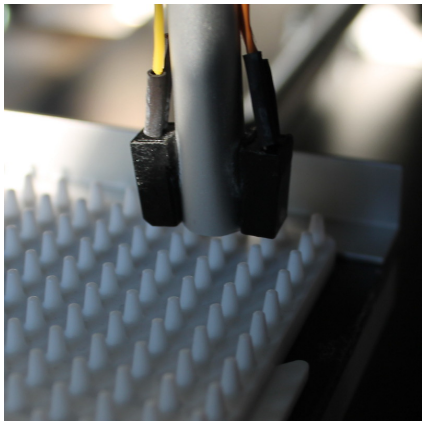
Vor der Kalibrierung sollte die *BeadBox* mithilfe des *Reset*-Knopfes (unten rechts) in die Nullposition gefahren werden und die Stiftplatte eingelegt sein.

Kalibrierung der Positionierungseinheit Damit die Positionierungseinheit funktioniert, müssen die Start- und Endwerte der Schrittmotoren bekannt sein. Für die Kalibrierung können die Schrittmotoren über die entsprechenden Schaltflächen des Hardwarekontrollprogramms (Abbildung 17, 2) schrittweise bewegt werden. Die Schrittmotoren können sich in x- und y-Richtung jeweils ca. 9200 Schritte weit bewegen, ohne an die Anschläge zu laufen.

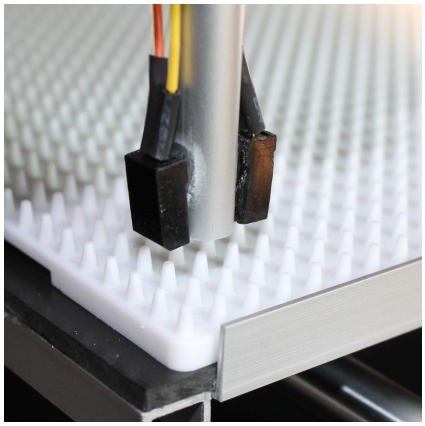
Damit man die Stiftplatte nach einem erfolgreichen Druckvorgang ohne Probleme entnehmen kann, wird auf jeder Seite der Stiftplatte ein Rand von zwei Stiften gelassen. So ist auch die Entnahme einer vollbedruckten Platte möglich, ohne die Stifte mit der Hand zu verschieben.



(a) Kalibrierungsschritt: oben links



(b) Kalibrierungsschritt: oben rechts



(c) Kalibrierungsschritt: unten links

Abbildung 18: Kalibrierung der Positionierungseinheit

Die Eckpunkte der Stiftplatte werden nacheinander angefahren. Begonnen wird mit dem Stift oben links. Mithilfe des Hardwarekontrollprogramms werden die Schrittmotoren beider Achsen so lange bewegt, bis sich das Platzierungsrohr genau über dem entsprechenden Stift befindet (Abbildung 18a). Ist die Position erreicht, müssen die Werte der Schrittmotoren in die Konfigurationsdatei übernommen werden. Ein Eintrag in der Konfigurationsdatei ist folgendermaßen aufgebaut:

```
<Punkt>=<x-Achsenschrittzaehler>, <y-Achsenschrittzaehler>
```

Zuerst kommt die Benennung des Eckpunktes (TopLeft für oben links, andere analog). Danach werden mit einem Komma getrennt die Werte des x-Achsenschrittmotorzählers und des y-Achsenschrittmotorzählers eingetragen.

Nachdem der erste Punkt kalibriert wurde, folgen die anderen Eckpunkte oben rechts (Abbildung 18b) und unten links (Abbildung 18c). Die entsprechenden Werte werden

ebenfalls in die Konfigurationsdatei eingetragen. Der Eintrag in der Konfigurationsdatei sollte nun wie folgt aussehen:

```
[PositioningCalibration]
TopLeft=<x-Achsenschrittzaehler>, <y-Achsenschrittzaehler>
TopRight=<x-Achsenschrittzaehler>, <y-Achsenschrittzaehler>
BottomLeft=<x-Achsenschrittzaehler>, <y-Achsenschrittzaehler>
```

Nach der Kalibrierung und Speicherung der Konfigurationsdatei können die kalibrierten Werte mithilfe der Positionskontrolle (Abbildung 17, 1) des Hardwaretestprogramms überprüft werden. Dazu muss das Hardwaretestprogramm neu gestartet werden.

Kalibrierung des Farbmagazins Das Kalibrieren des Farbmagazins ist notwendig, um den Nachlauf des Getriebemotors zu kompensieren. Allerdings muss der Kalibrierungsvorgang für das Farbmagazin in der Regel nicht vorgenommen werden. Das Magazin kann ebenfalls mit dem Hardwarekontrollprogramm gesteuert werden (Abbildung 17, 3).

Die in diesem Vorgang zu ermittelnden Werte dienen dazu, den Magazingetriebemotor kurz zurücklaufen zu lassen, damit die Magazinöffnung genau über der Öffnung des Vereinzelungsmechanismuses liegt. Dazu muss jedes Magazin einzeln angewählt werden und die Zeit, die für den ausgleichenden Rücklauf benötigt wird, experimentell bestimmt werden. Die Zeit sollte in einem Bereich von 20 - 50 ms liegen. Da sich das Magazin in zwei Richtungen bewegen kann, müssen dementsprechend auch zwei Ausgleichszeiten bestimmt werden. Die jeweilige Zeit wird in die Konfigurationsdatei eingetragen:

```
[MagazinCalibration]
ForwardAdjust<Nummer>=<Zeit in ms>
...
BackwardAdjust<Nummer>=<Zeit in ms>
```

Die Einträge befinden sich unter „MagazinCalibration“ in der Konfigurationsdatei. Für den Fall, dass das Magazin vorwärts läuft, beginnt der Eintrag mit „ForwardAdjust“. Anschließend kommt die Nummer des Magazins und dann die Ausgleichszeit in ms. Andernfalls beginnt der Eintrag entsprechend mit „BackwardAdjust“.

Konfiguration der Farben Die von der *BeadBox* verwendeten Perlenfarben werden ebenfalls in der Konfigurationsdatei eingetragen. Die Farbwerte befinden sich im Abschnitt „ColorPalette“ und haben folgenden Aufbau:

```
[ColorPalette]
Color00=#000000
Color01=#ffffff
...
Color13=#00ff00
```

Als erstes kommt das Schlüsselwort „Color“ mit angehängter Magazinnummer (immer zweistellig). Nach dem Gleichzeichen steht der RGB-Farbwert, der der Perlenfarben entspricht. Der Farbwert muss in Hex-Notation angegeben sein. Da das Farbmagazin maximal 14 verschiedene Farben unterstützt, dürfen auch nur maximal so viele Einträge vorhanden sein.

6.2.3. Druckvorgang

Im regulären Betrieb startet automatisch die *BeadBox*-Anwendungssoftware (siehe Abschnitt 5.5).

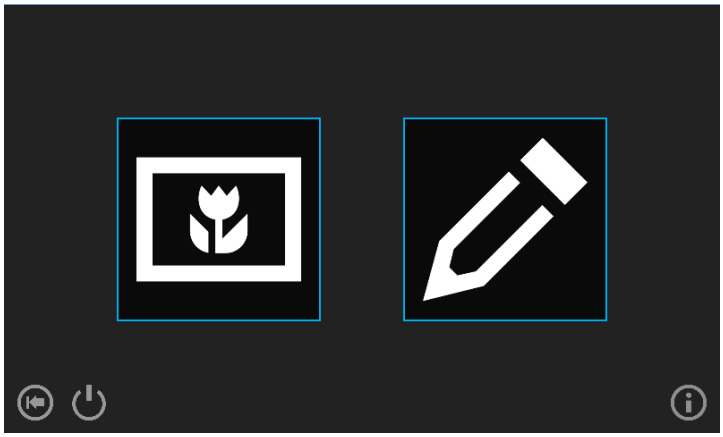


Abbildung 19: Hauptmenü

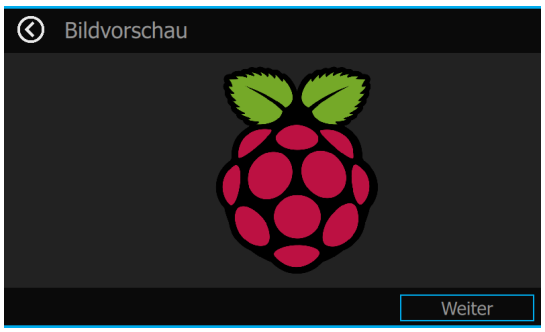
Im Hauptmenü (Abbildung 19) gibt es vier Schaltflächen, die durch eine Berührung aktiviert werden können. Mit der Schaltfläche links unten wird das Drucksystem in die Nullposition gesetzt. Die Schaltfläche daneben sorgt dafür, dass das Drucksystem in eine Standby-Position gefahren wird, in der es besser transportiert werden kann. Die Schaltfläche rechts unten zeigt ein Informationsfenster an.

In der Mitte des Hauptmenüs befinden sich die beiden Hauptschaltflächen, mit denen eine der beiden Druckvariationen (Rechts: Druck von Bildvorlage, Links: Druck von Malfläche) ausgewählt werden kann. Vor einem Druck sollte die Stiftplatte eingelegt sein und die Magazinbehälter aufgefüllte werden. Die einzelnen Farbbehälter sind mit der Perlenfarbe beschriftet.

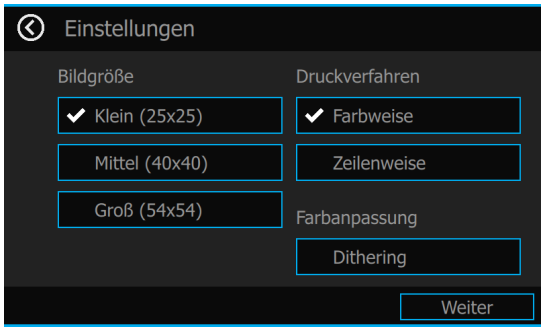
Druck von Bildvorlage Der Druck eines Bügelperlenbildes von einer Bildvorlage wird über die linke Schaltfläche gestartet. Der Klick auf diese Schaltfläche startet einen Wizard, der durch die notwendigen Schritte zur Fertigstellung des Bildes führt (Abbildung 20).

Als erstes muss eine Bilddatei als Vorlage ausgewählt werden. Dies geschieht im Bildauswahldialog (Abbildung 20a). Er zeigt ein Verzeichnis in Form einer Liste an. In der Liste sind wegen der Übersichtlichkeit nur Ordner und Bilddateien sichtbar. Bilddateien besitzen eine kleine Vorschau. Durch den Verzeichnisbaum kann auf bekannte Weise navigiert werden. Durch vertikales Wischen kann in der Liste gescrollt werden. Ein Klick auf einen Ordner öffnet diesen und ein Klick auf ein Bild zeigt eine Vorschau an (Abbildung 20b). Ist man mit der Auswahl des Bildes zufrieden, kommt man über „Weiter“ in den Einstellungsdialog (Abbildung 20c). Im Einstellungsdialog ist es möglich, die Größe des Bügelperlenbildes (klein, mittel, groß) und das Druckverfahren (farbweise, zeilenweise) auszuwählen. Außerdem kann die Farbanpassung auf Dithering umgestellt werden.

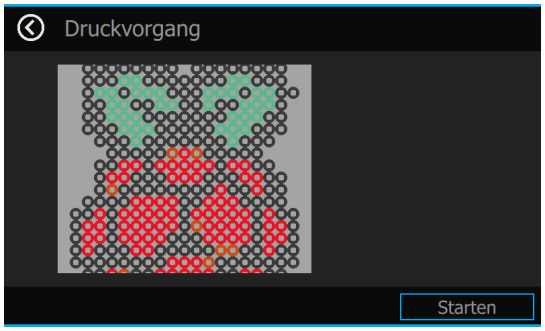
(a) Bildauswahl



(b) Bildauswahl



(c) Bildauswahl



(d) Druckfortschritt

Abbildung 20: Druckvorgang von einer Bildvorlage

Der letzte Dialog ist der Fortschrittsdialog (Abbildung 20d). Er zeigt zunächst eine Vorschau des fertigen Bügelperlenbildes an. Während des Druckvorgangs wird diese Ansicht kontinuierlich aktualisiert, sodass der aktuelle Druckfortschritt überwacht werden kann.

Druck von der Malfläche Über die rechte Schaltfläche im Hauptmenü wird ein Wizard gestartet, der durch die Schritte zum Drucken von der internen Malfläche führt (Abbildung 21).

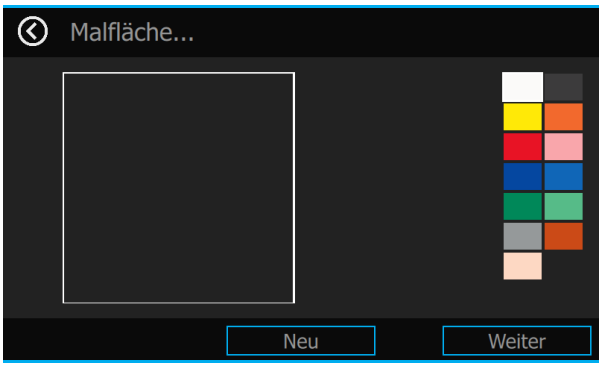


Abbildung 21: Malfläche

Über „Neu“ wird eine neue Malfläche erzeugt. Am rechten Rand befinden sich die verfügbaren Farben. Sie können durch einen Klick ausgewählt werden. Ist eine Farbe

gewählt worden, kann man im markierten Bereich zeichnen. Wenn das Ergebnis zufriedenstellend ist, kann mit der Schaltfläche „Weiter“ fortfahren werden. Die weiteren Schritte entsprechen denen des Drucks einer Bildvorlage. Erst werden die Druckeinstellungen vorgenommen und anschließend wird der Druck durchgeführt.

Sollte während des Druckvorganges ein Farbbehälter leer sein, wird der Druckvorgang pausiert und eine Fehlermeldung angezeigt. Über diese Meldung kann der Druckvorgang nach dem Auffüllen des Farbbehälters fortgesetzt werden. Es besteht auch die Möglichkeit einen Druckvorgang abzubrechen.

Nachdem der Druck abgeschlossen ist, kann die Stiftplatte entnommen und das Bild gebügelt werden.

7. Tests

Im Folgenden wird die Funktion der *BeadBox* anhand von einigen Tests und Benchmarks dargestellt. In diesen Tests wird das Verhalten des Systems im Fehlerfall (Fehler beim Vereinzeln, Fehler beim Absetzen) sowie die Performanz eines Druckvorgangs untersucht. Wichtige Messdaten dafür sind unter anderem der zurückgelegte Weg bei der Positionierung, die Anzahl der Farbwechsel und die Zeiten, die dafür benötigt werden. Des Weiteren ist die Dauer, die durchschnittlich zum Platzieren einer Perle benötigt wird, relevant. Mithilfe dieses Wertes kann man auf die Dauer von anderen Bildern schließen.

Für die Tests wurden Ausgangsbilder erstellt, die für den zu testenden Sachverhalt die aussagekräftigsten Ergebnisse lieferten. Jeder Test wurde zweifach ausgeführt. Die gemessenen Werte wurden gemittelt, um so bessere Vorhersagen treffen zu können.

Test 1: Vergleich der Druckverfahren Im ersten Test wurden die beiden verfügbaren Druckverfahren (farbweiser Druck, zeilenweiser Druck) miteinander verglichen. Hierbei ist vor allem die Anzahl der Farbwechsel, der Positionierungsweg und die Gesamtdauer des Druckvorgangs interessant. Mit dem farbweisen Druck soll ein Hin- und Herwechseln des Magazins vermieden und dadurch ein schnellerer Druckvorgang erreicht werden. Als Testbild dient ein Schwarz-Weiß-Schachbrettmuster (Abbildung 22). Die Größe des Testbildes beträgt 8 mal 8 Pixel. Diese Größe wurde gewählt, da sie vollständig mit einer Magazinfüllung gedruckt werden kann, sodass die Ergebnisse nicht durch ein leeres Magazin verfälscht werden.

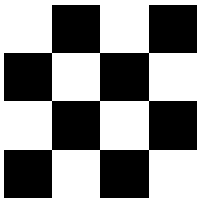


Abbildung 22: Testbild 1: Schachbrettmuster

Die Teststatistiken sind in Tabelle 2 aufgeführt. Aus den Ergebnissen ist ersichtlich, dass durch den farbweisen Druck die Anzahl der Farbwechsel im Vergleich zum zeilenweisen Druck erheblich reduziert werden konnte. Durch das Schachbrettmuster muss beim zeilenweisen Druck mehrfach die Farbe gewechselt werden. Eine Drehung des Magazins nimmt im Schnitt 1,3 Sekunden in Anspruch. Dadurch werden beim zeilenweisen Druck schon für die Farbwechsel 35,8 Sekunden benötigt. Dem gegenüber stehen 1,3 Sekunden für die eine Farbwechsel beim farbweisen Druck.

Beim zeilenweisen Druck ist der von der Positionierungseinheit zurückgelegte Weg zwar deutlich kürzer als beim farbweisen Druck, allerdings wird durchschnittlich mehr Zeit zum Anfahren einer Stiftposition benötigt. Dies liegt unter anderem daran, dass die Positionierungseinheit gleichzeitig die x- und die y-Achse bewegen kann. Dadurch schlägt sich der längere Weg nicht in der benötigten Zeit nieder.

Der farbweise Druck hat im Gegensatz zum zeilenweisen Druck öfter mit Fehlern beim Absetzen einer Perle zu kämpfen. Diese konnten jedoch alle durch eine Bewegung der Positionierungseinheit ausgeglichen werden. Die höhere Fehleranfälligkeit beim Absetzen liegt wahrscheinlich am größeren Positionierungsaufwand des farbweisen Drucks.

	Farbweise (8x8)	Zeilenweise (8x8)
Abgesetzte Perlen:	64	64
Farbwechsel:	1	28
Fehler bei der Vereinzelung:	0	0
Korrektur beim Absetzen:	15	5
Positionierungsweg (gesamt):	248,8 Pixel	105,5 Pixel
Positionierungsweg (durchschn.):	3,9 Pixel	1,6 Pixel
Zeit für Magazindrehung (gesamt):	1,3 s	35,8 s
Zeit für Positionierung (durchschn.):	0,5 s	0,7 s
Zeit für Vereinzelung (durchschn.):	1,4 s	1,6 s
Zeit pro Perle (durchschn.):	1,9 s	2,8 s
Dauer des Druckvorgangs:	120,8 s	177,6 s

Tabelle 2: Vergleich der Druckverfahren

Insgesamt schneidet der farbweise Druck im Vergleich zum zeilenweisen Druck wesentlich besser ab. Der farbweise Druck benötigt trotz der längeren Positionierungswege und der höheren Fehleranfälligkeit beim Absetzen ganze 57 Sekunden weniger Zeit für den Druck des Testbildes. Dies ist vor allem in den häufigen Farbwechseln begründet. In den weiteren Tests wird deshalb stets der farbweise Druck benutzt.

Test 2: Geschwindigkeit Der zweite Test misst das Verhalten der *BeadBox* bei zunehmender Bildgröße. Dieser Test wurde mit dem farbweisen Druck durchgeführt, da er in der Regel schneller als der zeilenweise Druck ist. Bei diesem Test ist hauptsächlich der Positionierungsweg, die Fehler beim Absetzen einer Perle und die Gesamtdauer ausschlaggebend. In diesem Test soll ermittelt werden, wie lange das Absetzen einer Perle im Schnitt dauert und wie sich die Größe des gedruckten Bildes auf die Gesamtdauer auswirkt. Die für diesen Test verwendeten Bilder sind in Abbildung 23 zu sehen.

Die drei unterschiedlich großen Testbilder sind zweifarbig (Schwarz-Weiß) und besitzen vier Streifen. Das erste Testbild hat eine Größe von vier mal vier, das zweite eine von acht mal acht und das dritte ein Größe von zwölf mal zwölf Pixeln. Da eine Magazinfüllung beim letzten Bild nicht ausreichte, wurden die Perlen während des Druckvorgangs aufgefüllt, damit die Ergebnisse nicht durch eine lange Unterbrechung (im Falle eines leeren Farbmagazins) verfälscht wurden.

Die Statistiken dieses Tests sind in Tabelle 3 dargestellt. Relativ naheliegend ist, dass mit zunehmender Größe des gedruckten Bildes auch der Positionierungsweg und die Positionierungsdauer zunimmt. Auch die Anzahl der Fehler beim Absetzen nimmt zu, je größer das Bild ist. Allerdings nimmt der prozentuale Anteil der Perlen, die nicht korrekt

(a) Testbild 2: 4x4 **(b)** Testbild 3: 8x8 **(c)** Testbild 4: 12x12

Abbildung 23: Testbilder 2 - 4: Streifenmuster

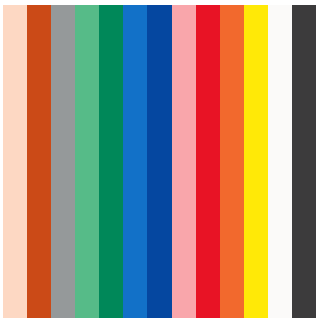
gesetzt wurden, leicht ab. Im Schnitt muss bei ungefähr 30 % der Perlen eine Korrektur durchgeführt werden.

	Bild (4x4)	Bild (8x8)	Bild (12x12)
Abgesetzte Perlen:	16	64	144
Farbwechsel:	1	1	1
Fehler bei der Vereinzelung:	0	0	0
Korrektur beim Absetzen:	6	19	43
Positionierungsweg (gesamt):	28,8 Pixel	195,1 Pixel	615,4 Pixel
Positionierungsweg (durchschn.):	1,7 Pixel	3,0 Pixel	4,3 Pixel
Zeit für Magazindrehung (gesamt):	1,3 s	1,3 s	1,3 s
Zeit für Positionierung (gesamt):	8,2 s	30,0 s	66,9 s
Zeit für Vereinzelung (gesamt):	23,6 s	79,8 s	212,6 s
Zeit pro Perle (durchschn.):	2,1 s	1,8 s	2,0 s
Dauer des Druckvorgangs:	33,8 s	114,8 s	281,1 s

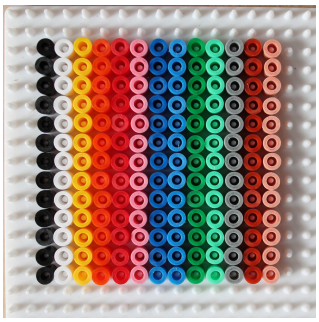
Tabelle 3: Druckvorgänge bei zunehmender Große

Insgesamt steigt die Gesamtdauer für einen Druckvorgang proportional mit der Anzahl der abgesetzten Perlen. Aus den Testergebnissen kann abgeleitet werden, dass zum Absetzen einer Perle im Schnitt 2,0 Sekunden benötigt werden. Daraus kann man die ungefähre Dauer eines Druckvorgangs bestimmen. Für ein 25 mal 25 Perlen großes Bild wird demnach theoretisch rund 21 Minuten gebraucht. Für die maximale bedruckbare Größe (54 mal 54 Perlen) ergibt sich eine theoretische Dauer von ca. 1 Stunde und 40 Minuten. In diesen Zeiten ist allerdings noch nicht mit einbezogen, dass die Magazine zwischendurch aufgefüllt werden müssen, weshalb ein Druckvorgang wahrscheinlich wesentlich länger dauern würde. Außerdem ist die Zeit, die für die Farbwechsel benötigt wird ebenfalls nicht berücksichtigt. Durch den farbweisen Druck, haben die Farbwechsel jedoch nur einen geringen Einfluss auf die Gesamtdauer und können deswegen vernachlässigt werden.

Test 3: Farbtest Da nun die wesentlichen Eigenschaften der *BeadBox* gemessen wurden, ist es interessant zu sehen, welche Ergebnisse aus den Ausgangsbildern entstehen. Das Bild in Abbildung 24a ist ein Testbild, mit dem das Farbmagazin geprüft wird. Es enthält alle Farben die von der *BeadBox* gedruckt werden können und ist dreizehn mal dreizehn Pixel groß. Rechts daneben ist das von der *BeadBox* aus dem Testbild erstellte Bügelperlenbild zu sehen (Abbildung 24b). Zu beachten ist, dass das Bügelperlenbild vertikal gespiegelt ist, da Bügelperlenbilder auf der Rückseite gebügelt werden. Der Druckvorgang wurde mit dem farbweisen Verfahren durchgeführt.



(a) Ausgangsbild



(b) Bügelperlenbild

Abbildung 24: Testbild 5: Farbentest

Dieser Druckvorgang hat rund 6 Minuten gedauert. Dabei gab es zwei Fehler bei der Vereinzelung, die jedoch schnell behoben werden konnten. 49 mal musste nach dem Absetzen einer Perle korrigiert werden. Dieser Wert liegt wie bei den vorherigen Tests bei leicht unter 30 % der abgesetzten Perlen.

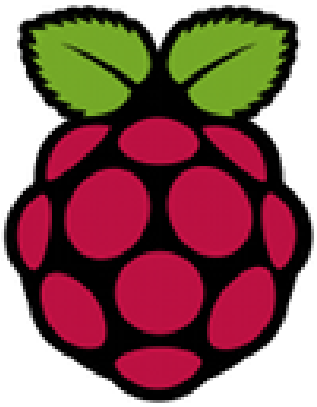
	Farbtest (13x13)
Abgesetzte Perlen:	169
Farbwechsel:	12
Fehler bei der Vereinzelung:	2
Korrektur beim Absetzen:	49
Zeit pro Perle (durchschn.):	2,1 s
Dauer des Druckvorgangs:	6 min

Tabelle 4: Statistik des Farbtests

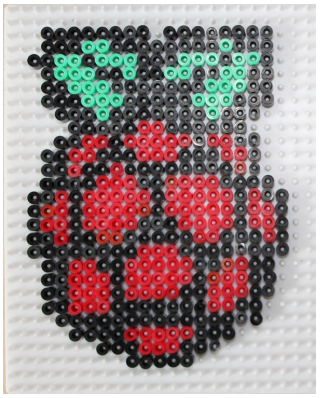
Das Ergebnis dieses Druckvorgangs entspricht dem Ausgangsbild und zeigt das alle Farben aus dem Magazin entnommen werden können. Somit ist nachgewiesen, dass die *BeadBox* im Prinzip voll einsatzfähig ist.

Anwendungsbeispiel Im letzten Test wird ein realistisches Anwendungsbeispiel durchgeführt. Dieses Beispiel soll alle Funktionen der *BeadBox* ausreizen und so die Tauglichkeit für den produktiven Einsatz demonstrieren. Als Ausgangsbild dient das Raspberry

Pi Logo (Abbildung 25a). Das Bild ist 570 mal 720 Pixel groß. Aus dem Ausgangsbild soll ein maximal 25 mal 25 Pixel großes Bügelperlenbild entstehen. Der Druck wird ebenfalls farbweise durchgeführt.



(a) Ausgangsbild



(b) Bügelperlenbild

Abbildung 25: Anwendungsbeispiel: Raspberry Pi Logo

Das erzeugte Bügelperlen Bild ist in Abbildung **??** zu sehen. Es besteht aus insgesamt 397 Bügelperlen und ist 9,5 cm mal 12 cm groß. Das Bügelperlenbild zeigt trotz der geringen Auflösung alle Merkmale des Ausgangsbildes und erfüllt die Erwartungen in vollem Umfang. Für den Druck wurden vier verschiedenfarbige Bügelperlen benötigt: *schwarz*, *rot*, *rotbraun* und *hellgrün*. Diese Farben wurden mithilfe eines einfachen Farbangleichs aus dem Ursprungsbild ermittelt.

	Raspberry Pi Logo
Abgesetzte Perlen:	397
Farbwechsel:	3
Fehler bei der Vereinzelung:	10
Korrektur beim Absetzen:	128
Zeit pro Perle (durchschn.):	4,2 s
Dauer des Druckvorgangs:	27,6 min

Tabelle 5: Statistik des Raspberry Pi Logos

Der Druckvorgang wurde mehrfach unterbrochen. Hauptsächlich weil die Magazine leer waren. Zweimal musste allerdings eine Verklemmung im Vereinzelungsmechanismus von Hand behoben werden. Der Druckvorgang nahm rund 28 Minuten in Anspruch. Dies liegt zum Teil daran, dass die Magazine zwischenzeitlich aufgefüllt werden mussten. Das gewählte Anwendungsbeispiel zeigt eindeutig, dass die *BeadBox* für den produktiven Einsatz geeignet ist und auch größere Bilder erfolgreich erstellen kann.

8. Fazit

Die in dieser Arbeit entwickelte *BeadBox* zur automatisierten Erstellung von Bügelperlenbildern erfüllt einen Großteil der am Anfang aufgestellten Anforderungen. Das Drucksystem ermöglicht ein genaues Positionieren, um eine Perle korrekt abzusetzen. Ebenfalls kann Dank des Revolver-Farbmagazins beim Drucken auf mehrere Bügelperlenfarben zurückgegriffen werden. Der kompakte Aufbau aus Aluminium und Kunststoff sorgt für eine ausreichende Robustheit und eine einfache Transportierbarkeit.

Bei der Software wurden alle vorgenommenen Ziele erreicht und teilweise sogar noch überschritten. Die Bedienung erfolgt über eine einfache und übersichtliche Oberfläche, die keine Fehlbedienung zulässt. Der Nutzer wird über aufgetretene Fehler durch Meldungen informiert. Währenddessen wird der Druckvorgang unterbrochen. Außerdem wird der Druckfortschritt durch eine Live-Vorschau visualisiert. Ein Feedback an den Benutzer ist also gewährleistet. Durch das fehlertolerante serielle Übertragungsprotokoll ist eine sichere Kommunikation sowie eine sichere Synchronisation zwischen Hardware und Software erreicht. Die Software ist durch die Verwendung von Qt vollständig plattformunabhängig und auch auf dem Raspberry Pi lauffähig.

Leider konnten aufgrund von fertigungsbedingten Ungenauigkeiten keine Hama-Mini-Perlen verwendet werden. Stattdessen mussten die größeren Midi-Perlen eingesetzt werden, was nur eine geringere Auflösung der erstellbaren Bilder zulässt. Die Ungenauigkeiten sind auf die manuelle Fertigung der Bauteile zurückzuführen. Durch eine maschinelle Fertigung der Bauteile wären diese Probleme höchstwahrscheinlich vermeidbar gewesen. Schwierigkeiten, die bei der Entwicklung von Maschinenbauteilen auftraten, mussten durch kreative Lösungen kompensiert werden.

Aufgrund des Wechsels der Bügelperlengröße konnten die komfortablen großen Perlenbehälter im Farbmagazin nicht umgesetzt werden. Durch die größeren Perlen traten Platzprobleme auf. Konzeptionell würde der in Abschnitt 4.4 vorgestellte zweite Lösungsversuch funktionieren. Das Lösen der Platzprobleme hätte allerdings einen großangelegten Umbau des Drucksystems zur Folge gehabt. Aus Zeitgründen war dieser Umbau jedoch nicht möglich.

Bei der *BeadBox* wurde, trotz der fertigungsbedingten Ungenauigkeiten der Hardware, für ein fehlertolerantes Verhalten gesorgt. Dies wurde u.a. durch Sensoren, die das Absetzen der Perle überprüfen, erreicht. Etwaige Fehler, wie beispielsweise eine fehlerhaft abgesetzte Perle, werden erkannt und per Software behoben.

Das Beheben von Softwarefehlern im Drucksystem und im Arduino-Code nahm während der Entwicklung relativ viel Zeit in Anspruch. Oft waren Kleinigkeiten der Auslöser für einen Komplettabsturz des System. So sorgte z.B. eine unglückliche Kombination aus serieller Kommunikation und Timer-Interrupt sporadisch für einen Komplettabsturz des Arduino. Abgesehen davon war die serielle Kommunikation unerwartet sehr fehleranfällig. Ab und zu gingen Nachrichten verloren, weswegen das ursprüngliche einfache Übertragungsprotokoll durch die jetzige Version ersetzt wurde. Die nun verwendete Protokollimplementation kann solche Übertragungsfehler beheben. Weiterhin hatte ein Speicherüberlauf im Arduino zur Folge, dass der Arduino-Bootloader überschrieben wurde. Dadurch war der Arduino nicht mehr über USB ansprechbar. Dies konnte jedoch durch

ein Neuprogrammieren des Bootloaders mithilfe eines externen Programmiergerätes repariert werden.

Alles in allem ist mit der *BeadBox* in der jetzigen Version ein Drucken von beliebigen Ausgangsbildern in mehreren Größen möglich. Um ein gutes Ergebnis zu erhalten, sollten einfache plakative Bilder, wie beispielsweise Logos oder Icons, gewählt werden.

Die aktuell vorhandenen Bildverarbeitungsfilter (Skalierung, Farbanpassung/Dither) sind nicht in der Lage, die Bildinformationen von komplexen Bildern für die geringe Auflösung der Bügelperlenbilder auf angemessene Weise vorzubereiten. Hier bietet sich Potential für zukünftige Erweiterungen. Es wäre beispielsweise möglich, die Bilder durch neuronale Netze aufzubereiten. Die neuronalen Netze könnten angelernt werden die wesentlichen Bildinformationen zu erkennen und herauszustellen. So könnte man auch komplexe Bilder in vereinfachter Form (eventuell comic-artig) drucken.

Eine weitere Möglichkeit wäre es, die Ausgangsbilder zu vektorisieren, sie also in Vektorgraphiken zu überführen. Vektorgraphiken können beliebig skaliert werden, sodass sie leichter an die Auflösung der Bügelperlenbilder angepasst werden können. Allerdings ist die Vektorisierung von Rastergraphiken (normale Bilder) sehr schwierig und es können sehr große und komplexe Vektorgraphiken bei diesem Vorgang entstehen. Aus Zeitgründen konnte jedoch nicht weiter in diese Richtung geforscht werden.

Zukünftig könnte auch eine Verbesserung an den Druckverfahren durchgeführt werden, um den Druckvorgang zu beschleunigen. Der farbweise Druck liefert zwar schon gute Ergebnisse, er benutzt allerdings nicht den optimalen Weg bei der Positionierung. Hier wäre vorstellbar, durch Entscheidungsbäume oder neuronale Netze den optimalen Druckablauf zu finden, der die Farbwechsel und die Positionierungsdauer minimiert.

Abschließend bleibt noch zu sagen, dass die Entwicklung der *BeadBox* trotz aller Schwierigkeiten und Probleme erfolgreich war. Die *BeadBox* bildet ein informatisches System, welches dem Anwender die Erstellung von Bügelperlenbildern erlaubt und dabei gleichzeitig die Fähigkeiten informatischer Systeme veranschaulicht.

A. Zeichnungen

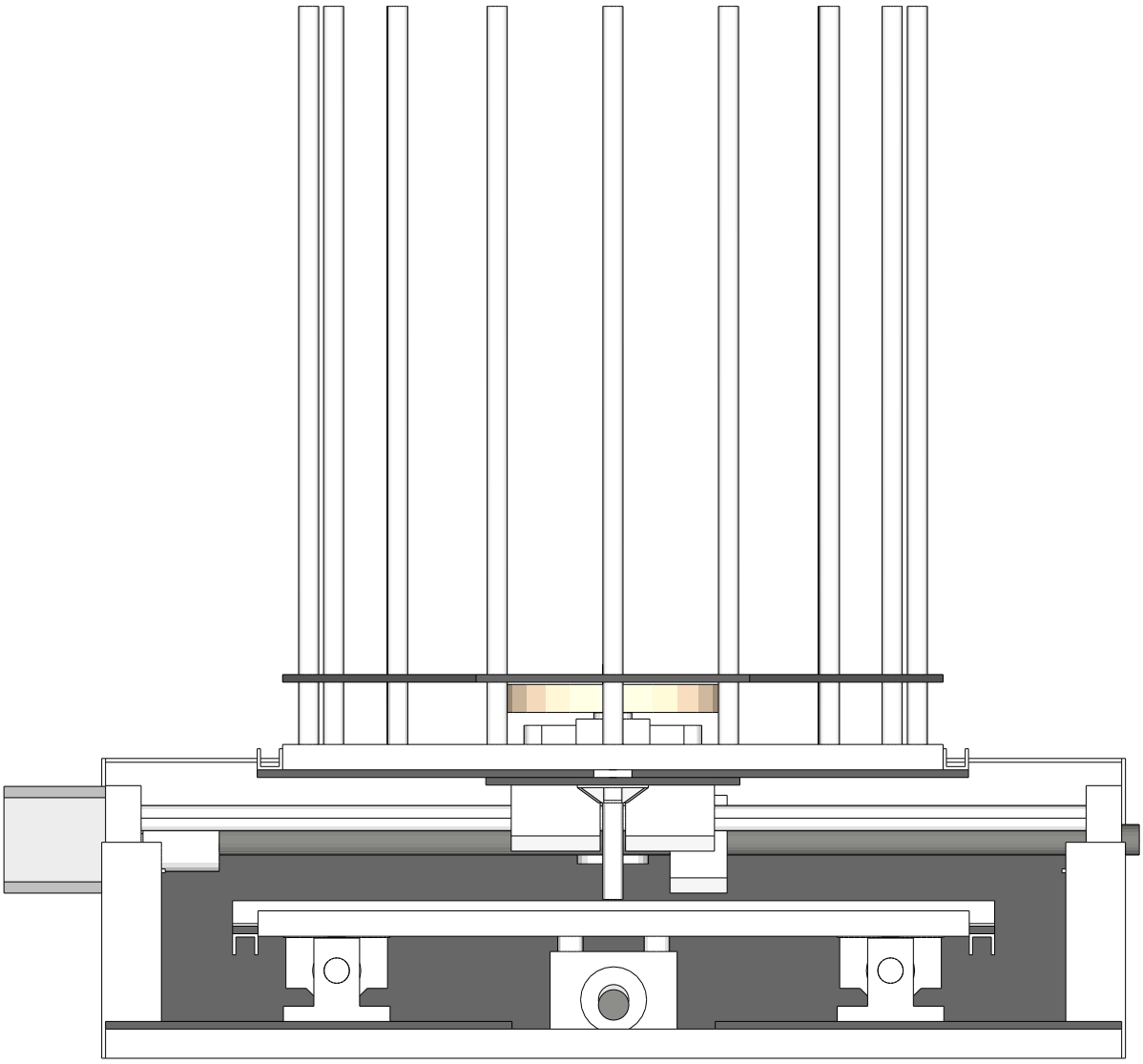


Abbildung 26: Hardwareaufbau der *BeadBox* - Ansicht Vorne

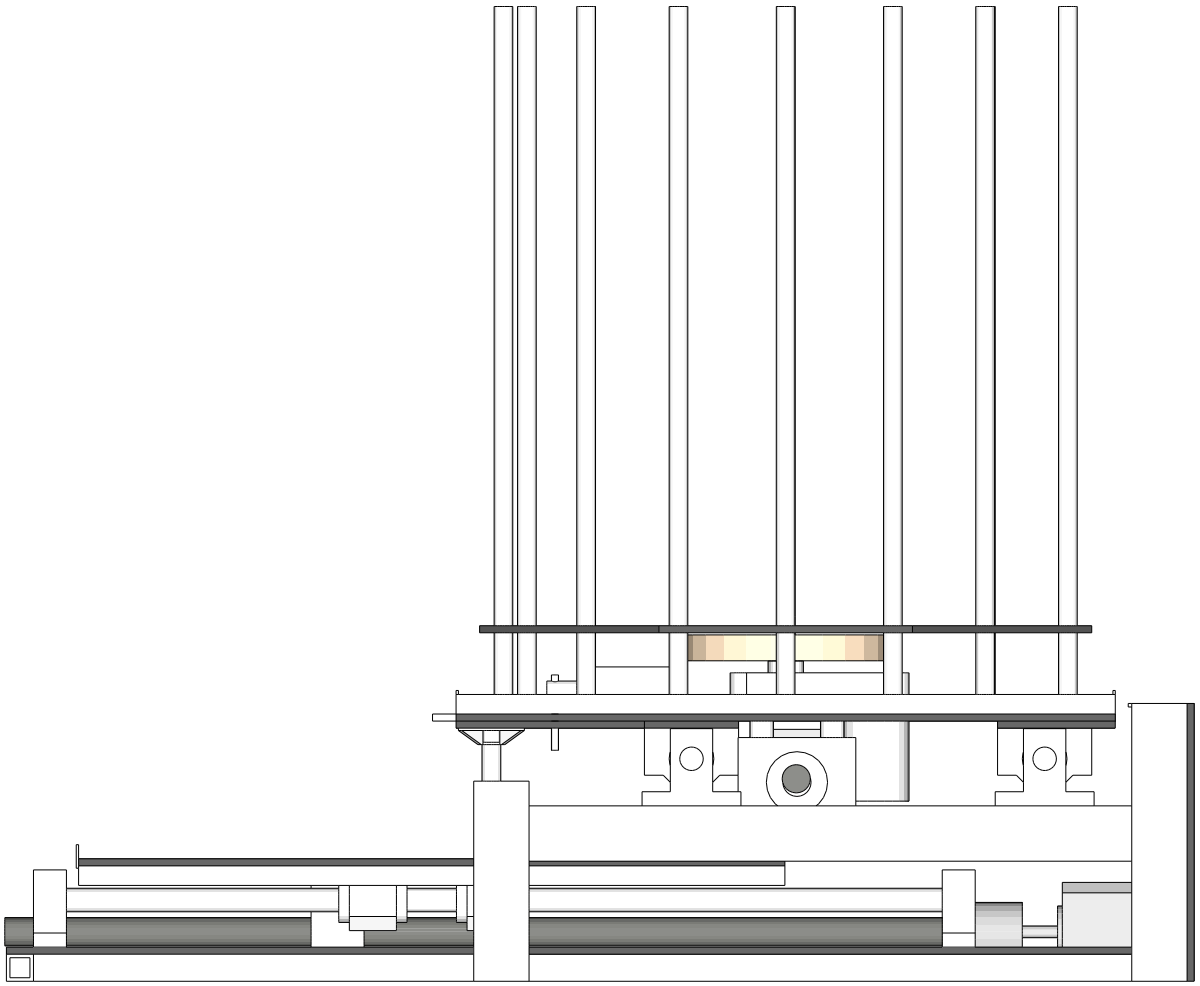


Abbildung 27: Hardwareaufbau der *BeadBox* - Ansicht Rechts

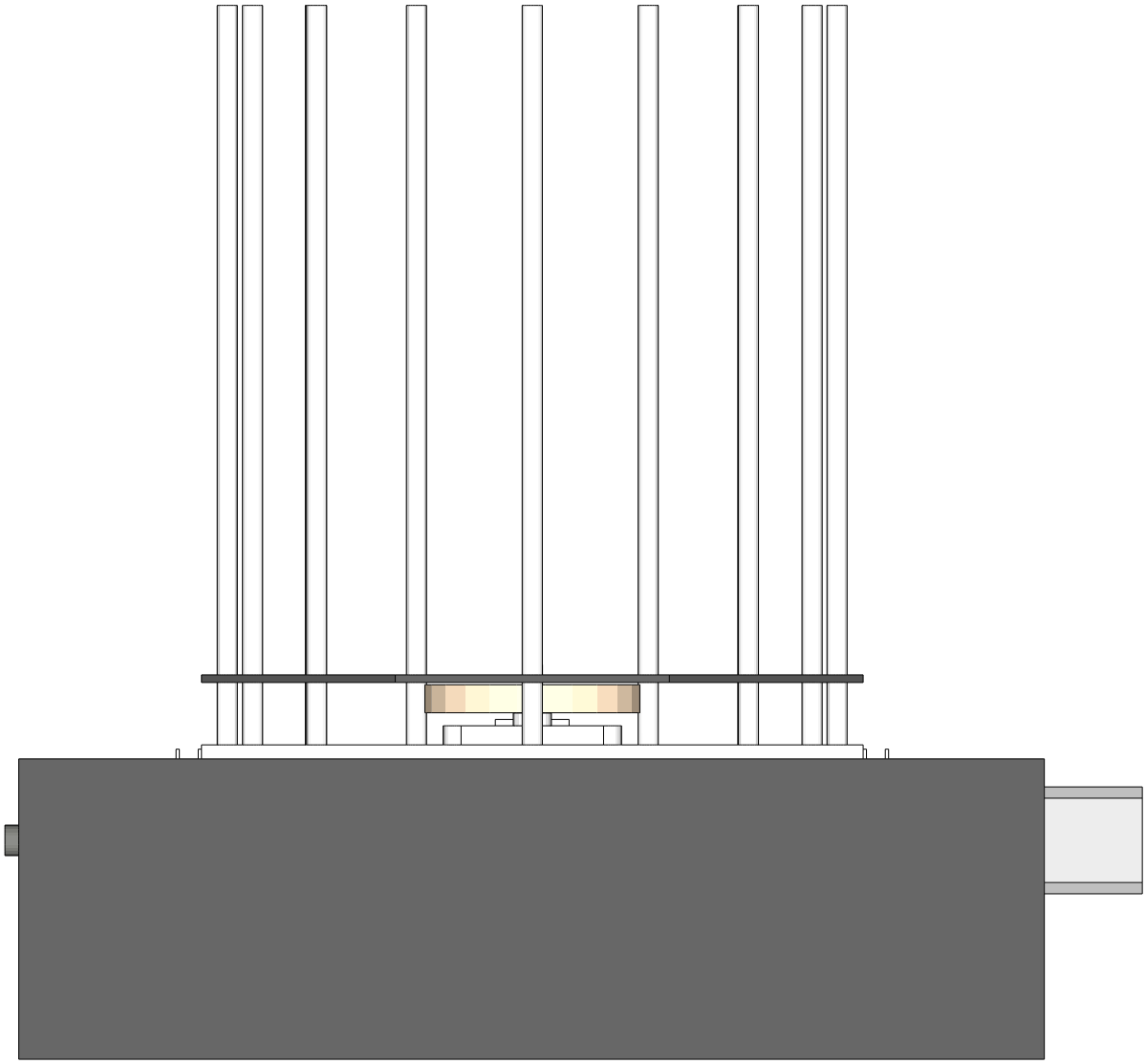


Abbildung 28: Hardwareaufbau der *BeadBox* - Ansicht Hinten

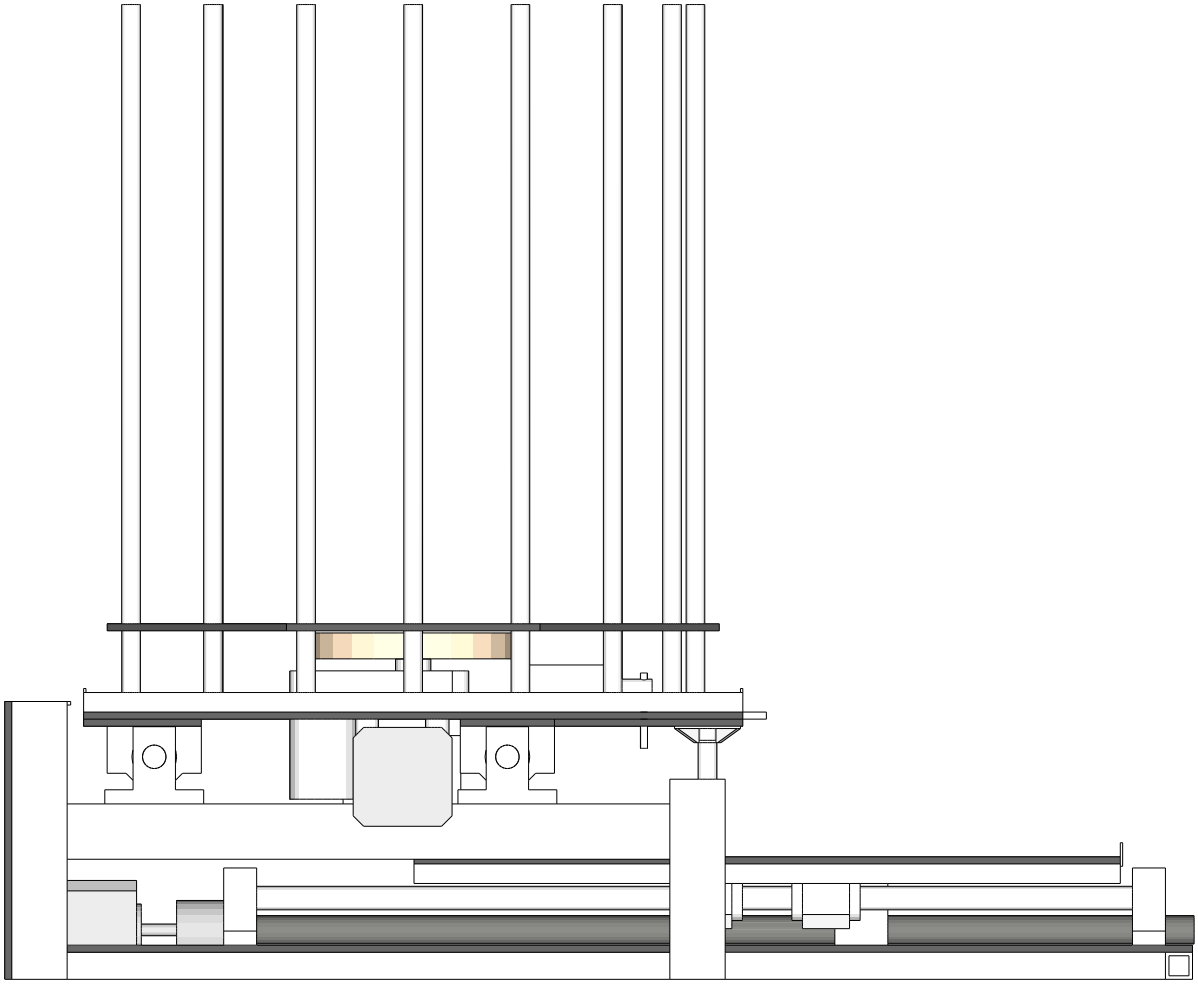


Abbildung 29: Hardwareaufbau der *BeadBox* - Ansicht Links

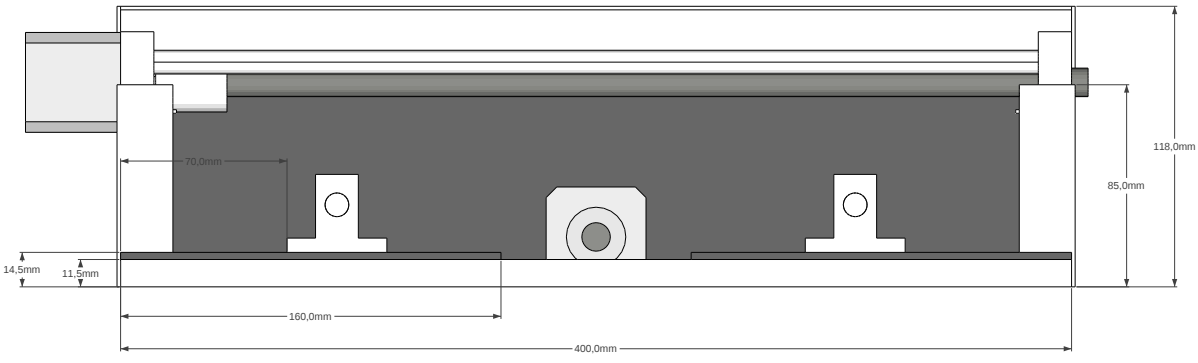


Abbildung 30: Rahmen der *BeadBox* - Ansicht Vorne

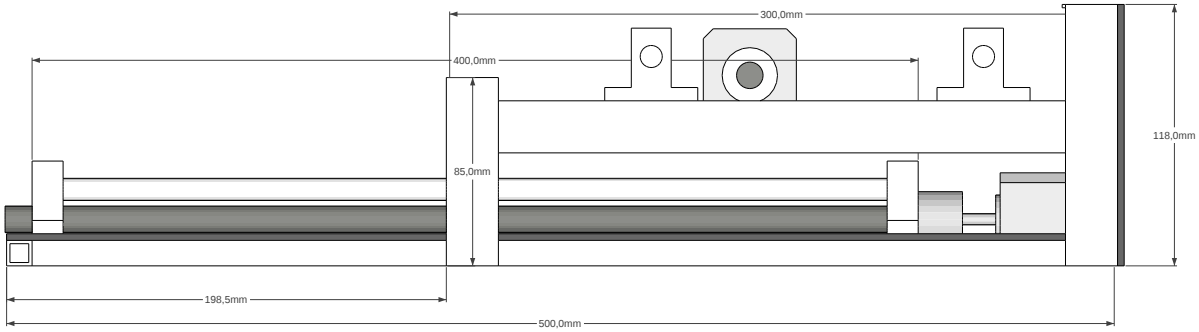


Abbildung 31: Rahmen der *BeadBox* - Ansicht Rechts

Abbildung 32: Rahmen der *BeadBox* - Ansicht Hinten

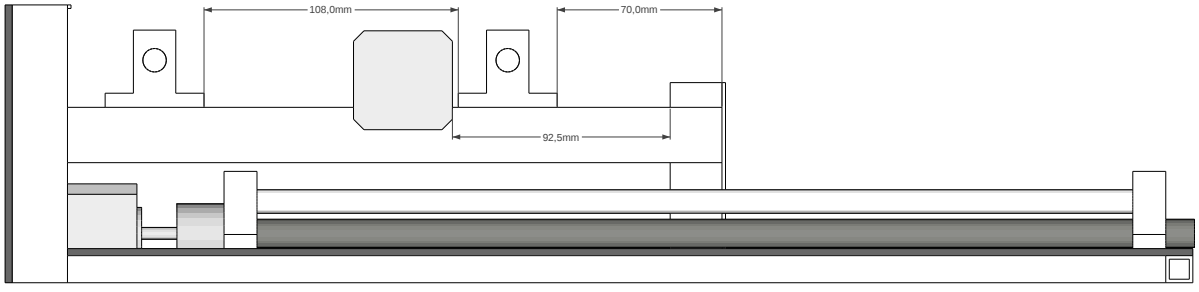


Abbildung 33: Rahmen der *BeadBox* - Ansicht Links

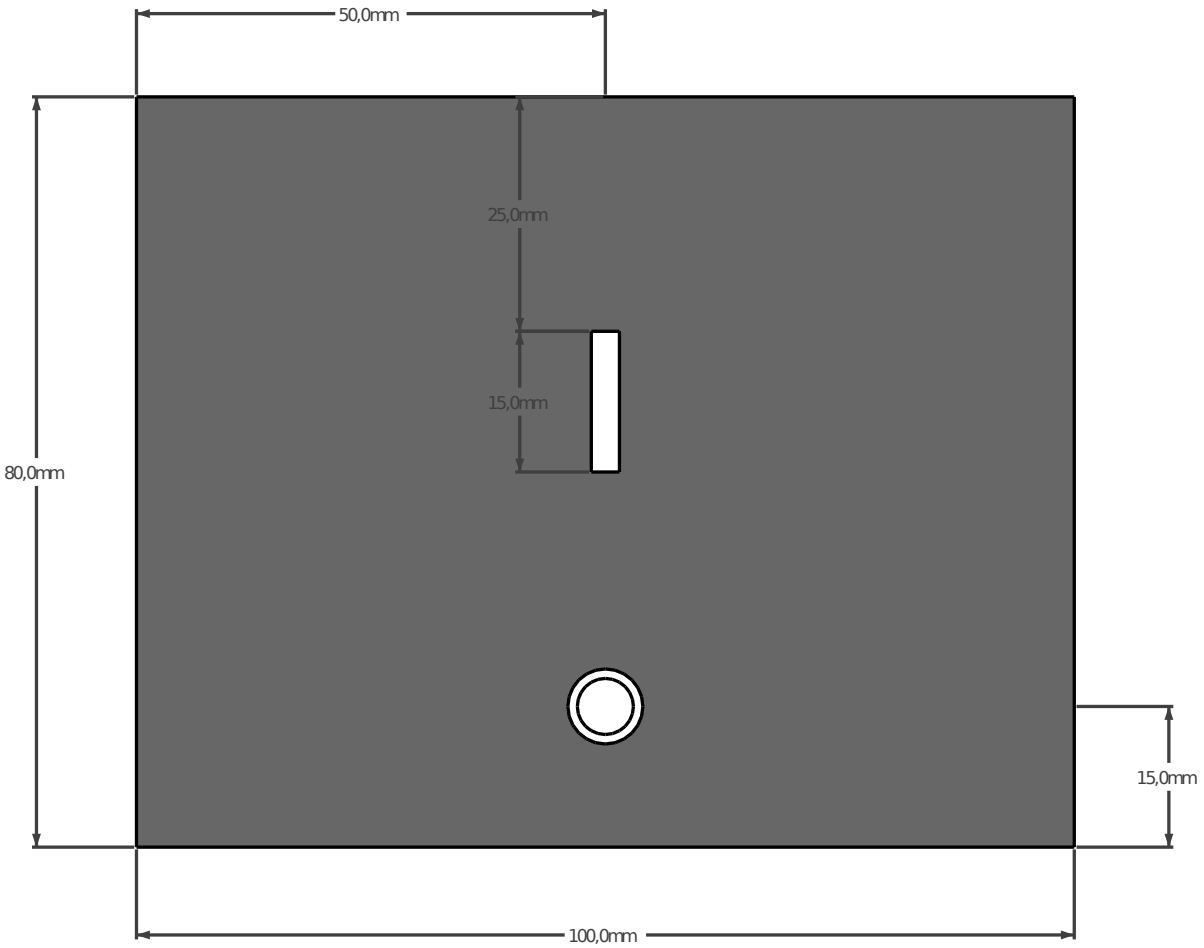


Abbildung 34: Vereinzelung - Ansicht Oben

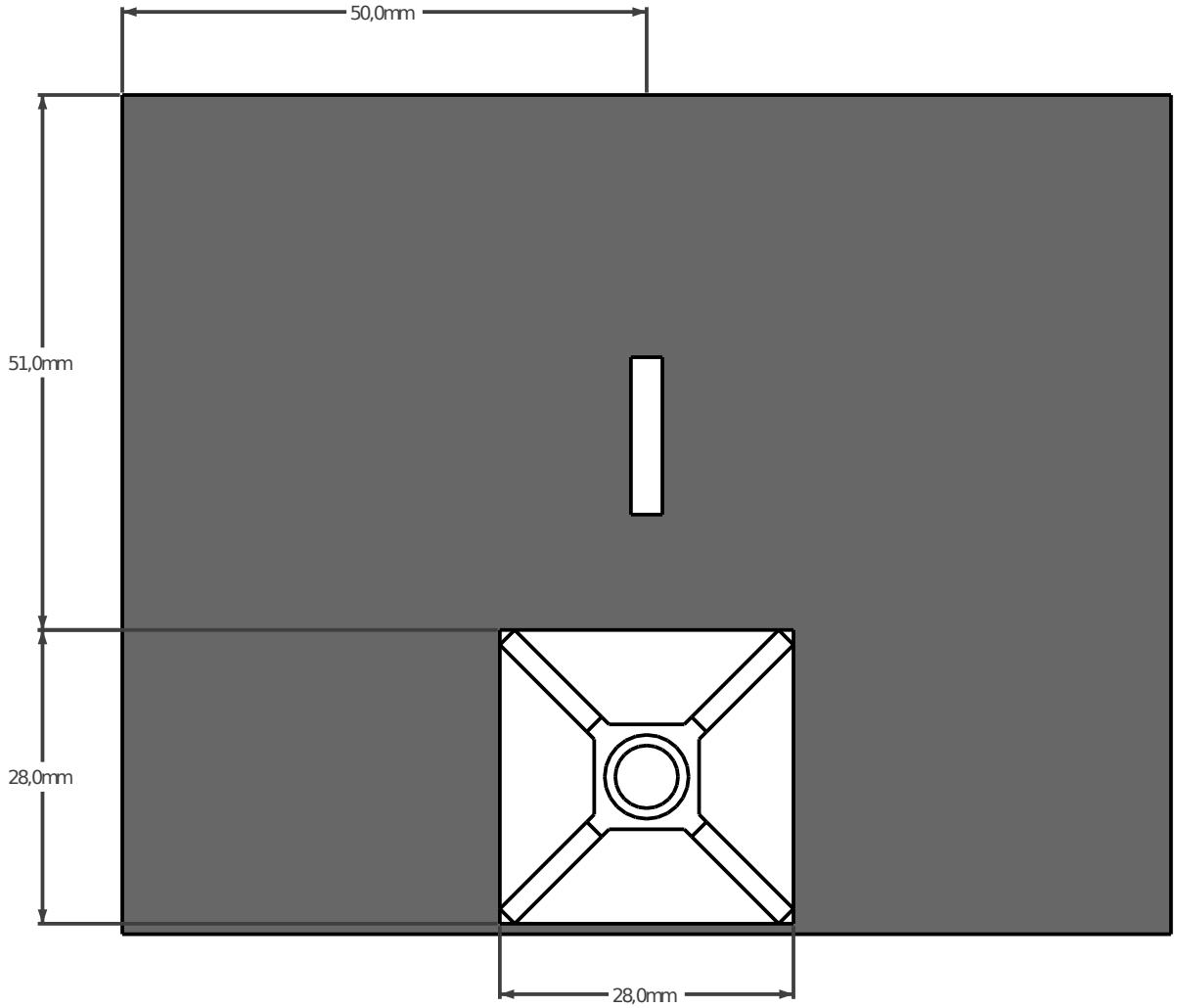


Abbildung 35: Vereinzelung - Ansicht Unten

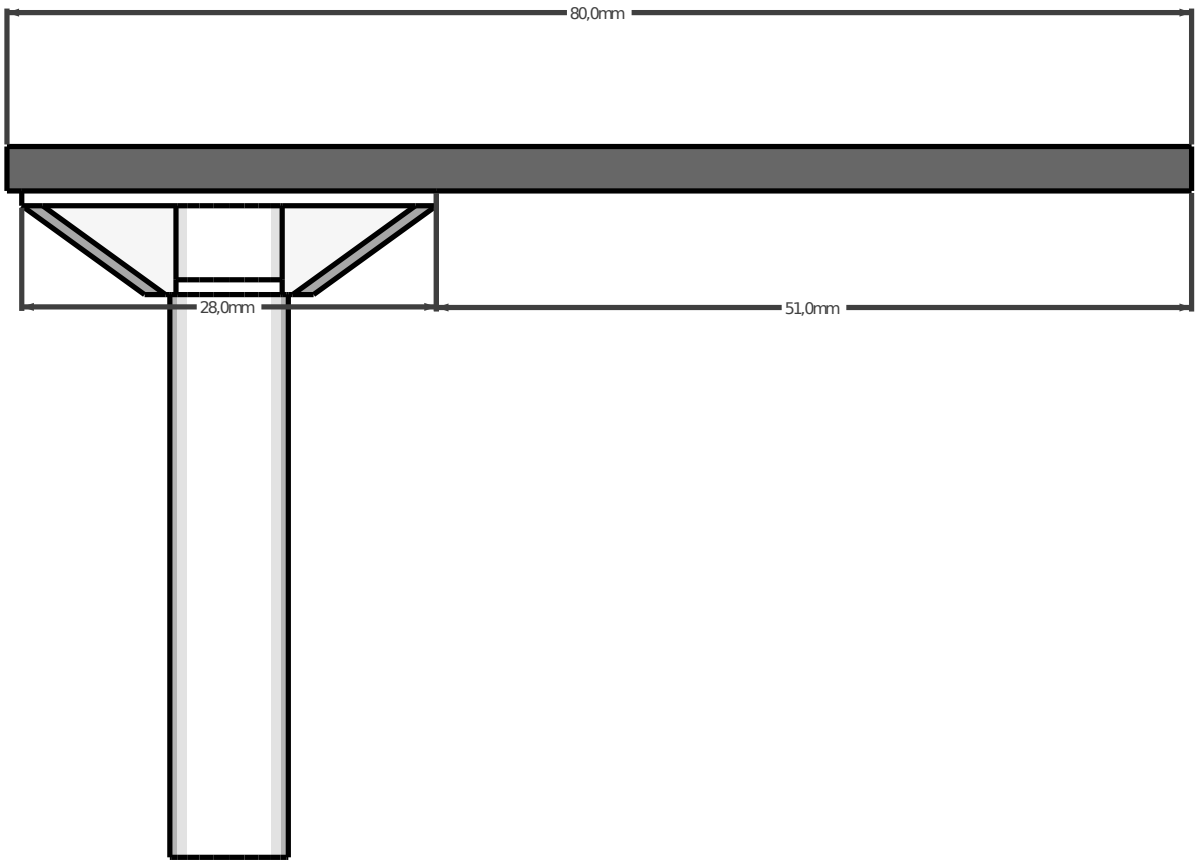


Abbildung 36: Vereinzelung - Ansicht Rechts

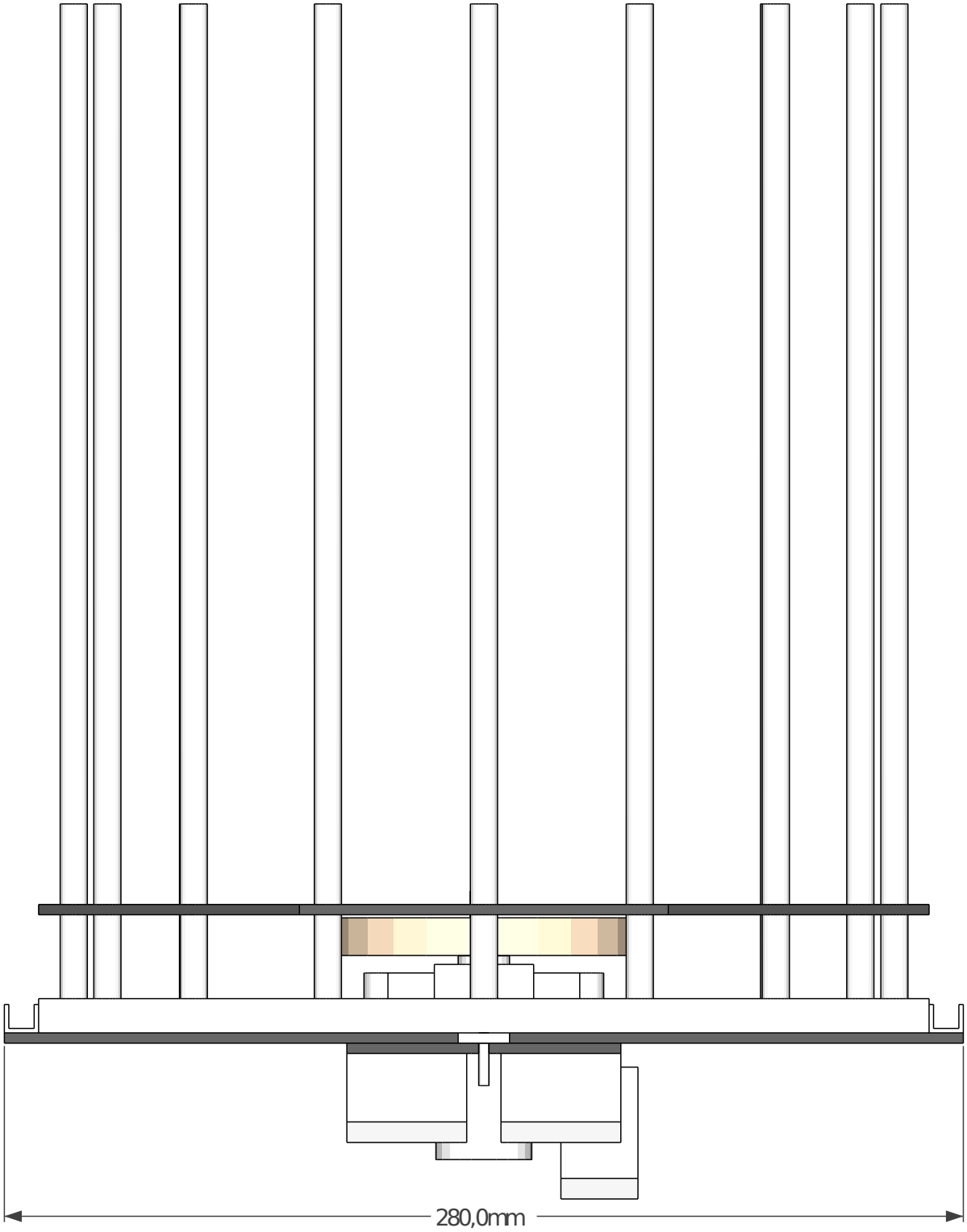


Abbildung 37: X-Schlitten - Ansicht Vorne

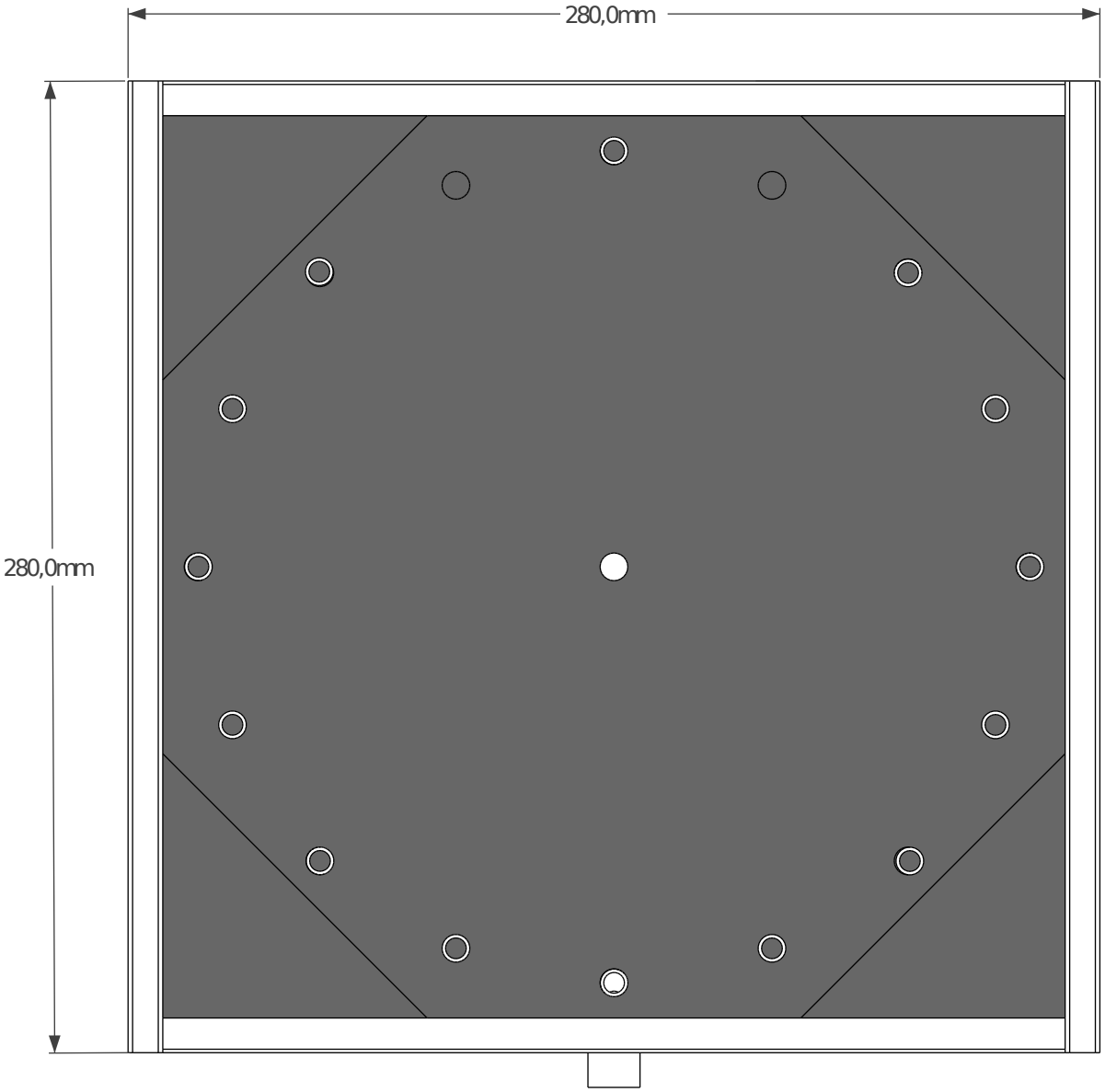


Abbildung 38: X-Schlitten - Ansicht Oben

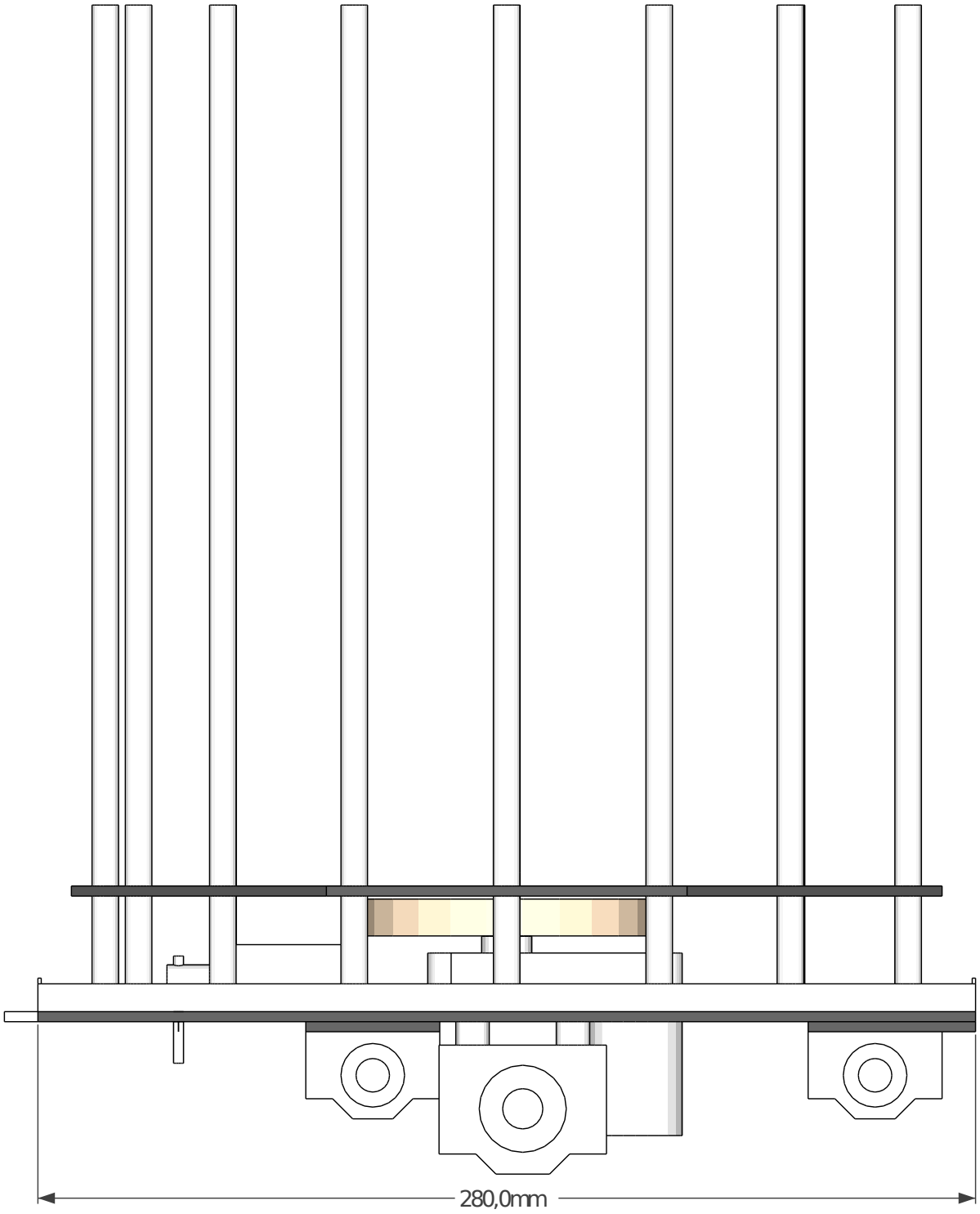


Abbildung 39: X-Schlitten - Ansicht Rechts

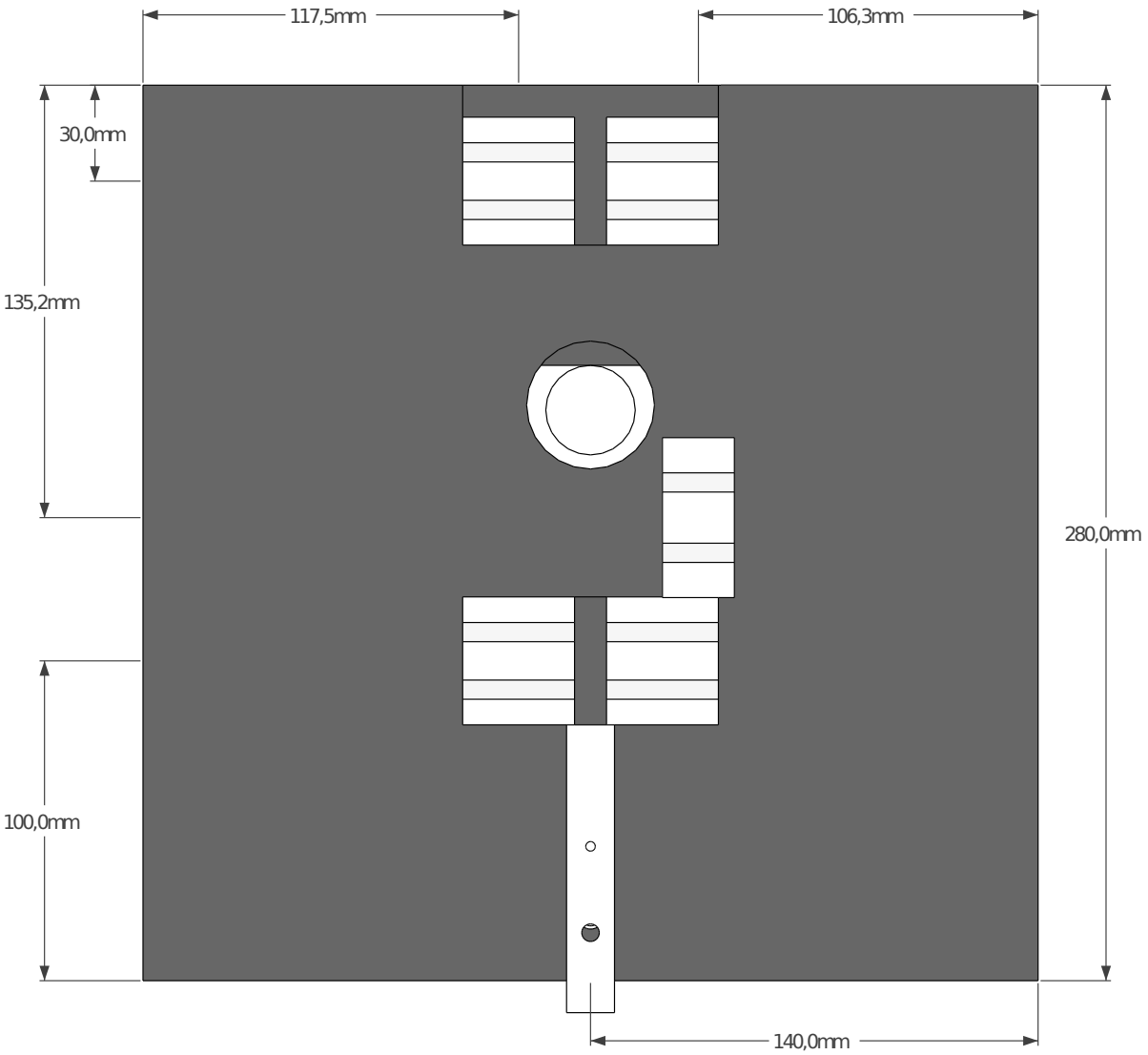


Abbildung 40: X-Schlitten - Ansicht Unten

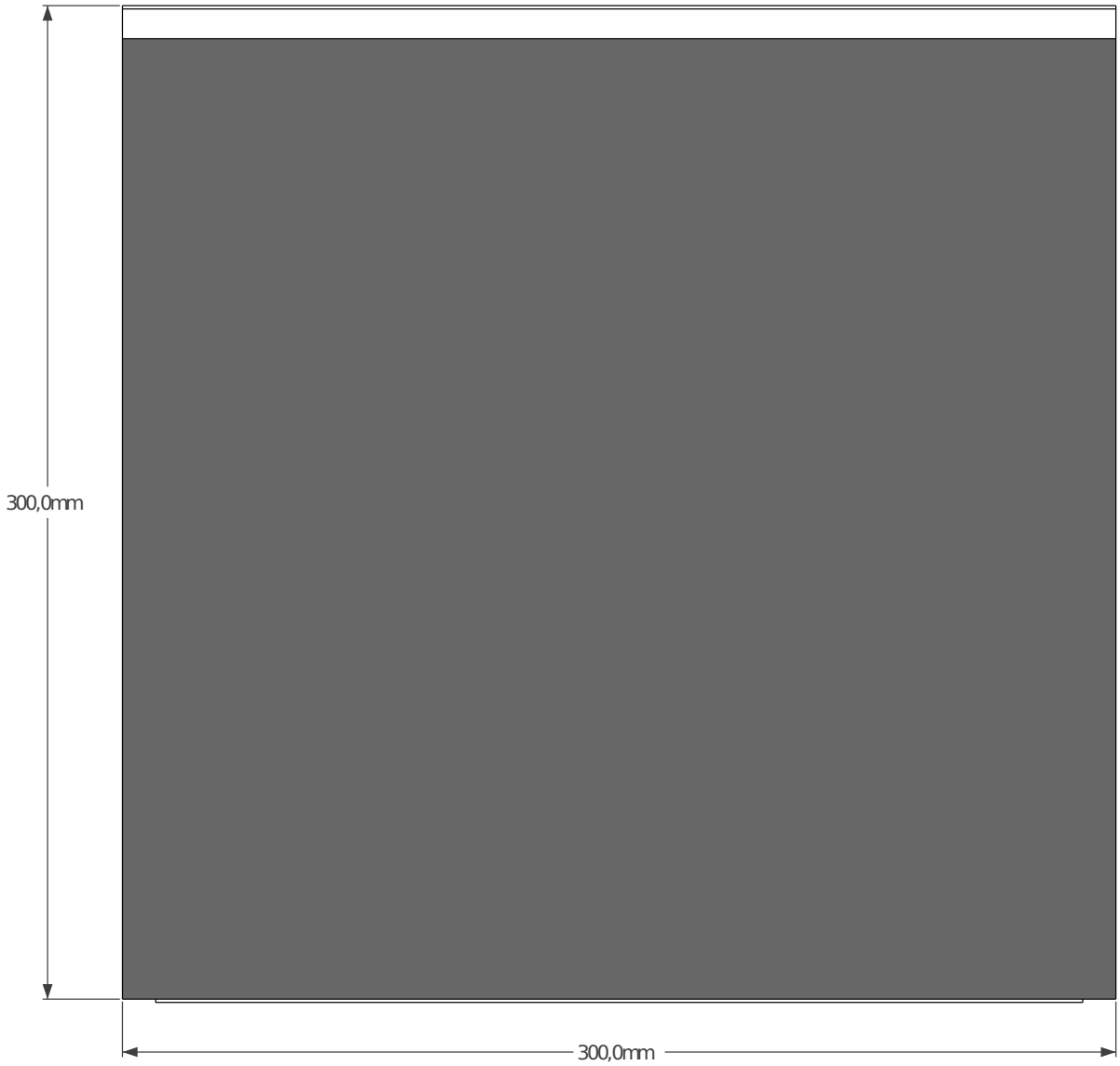


Abbildung 41: Y-Schlitten - Ansicht Oben

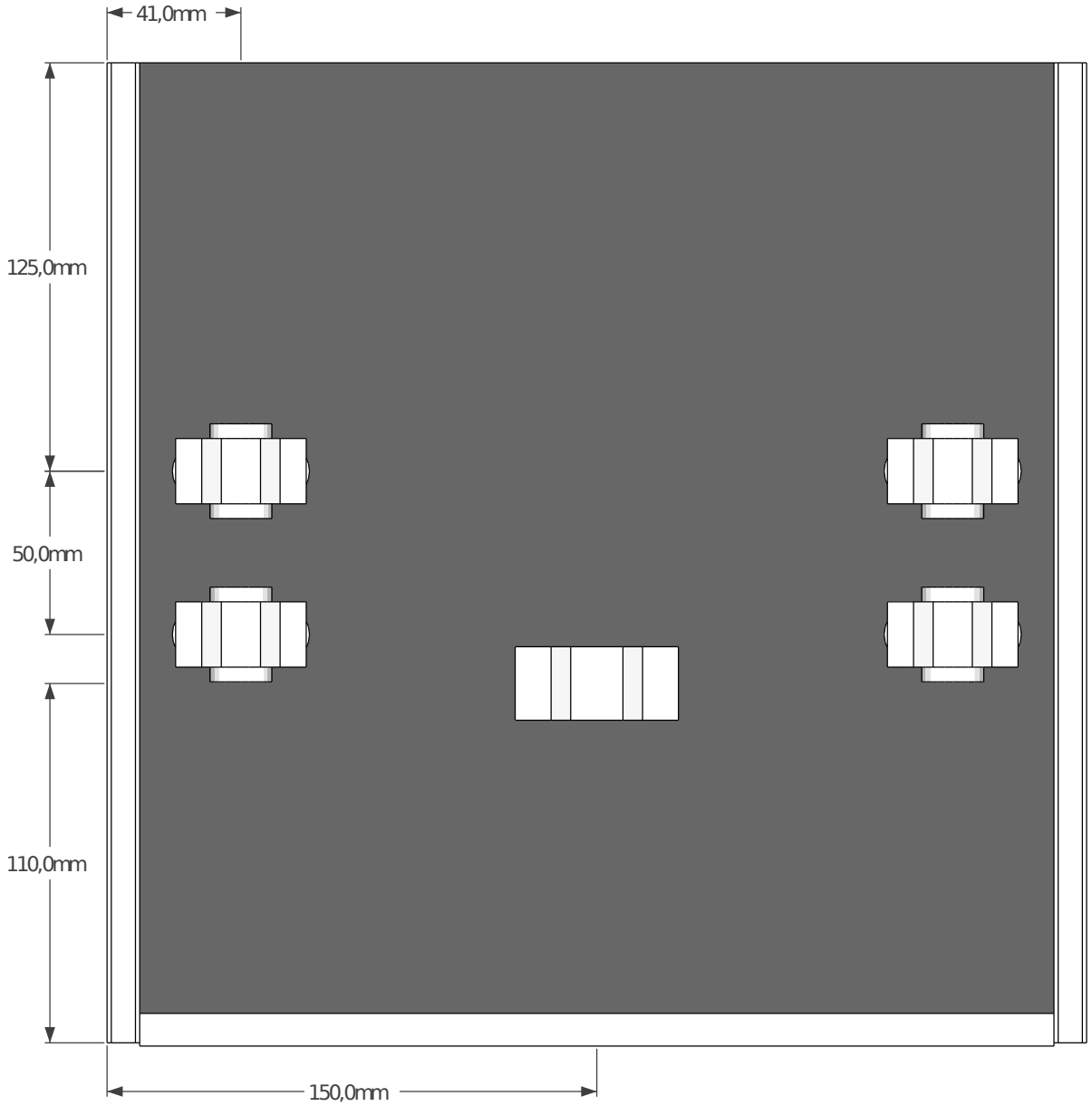


Abbildung 42: Y-Schlitten - Ansicht Unten

B. Schaltpläne

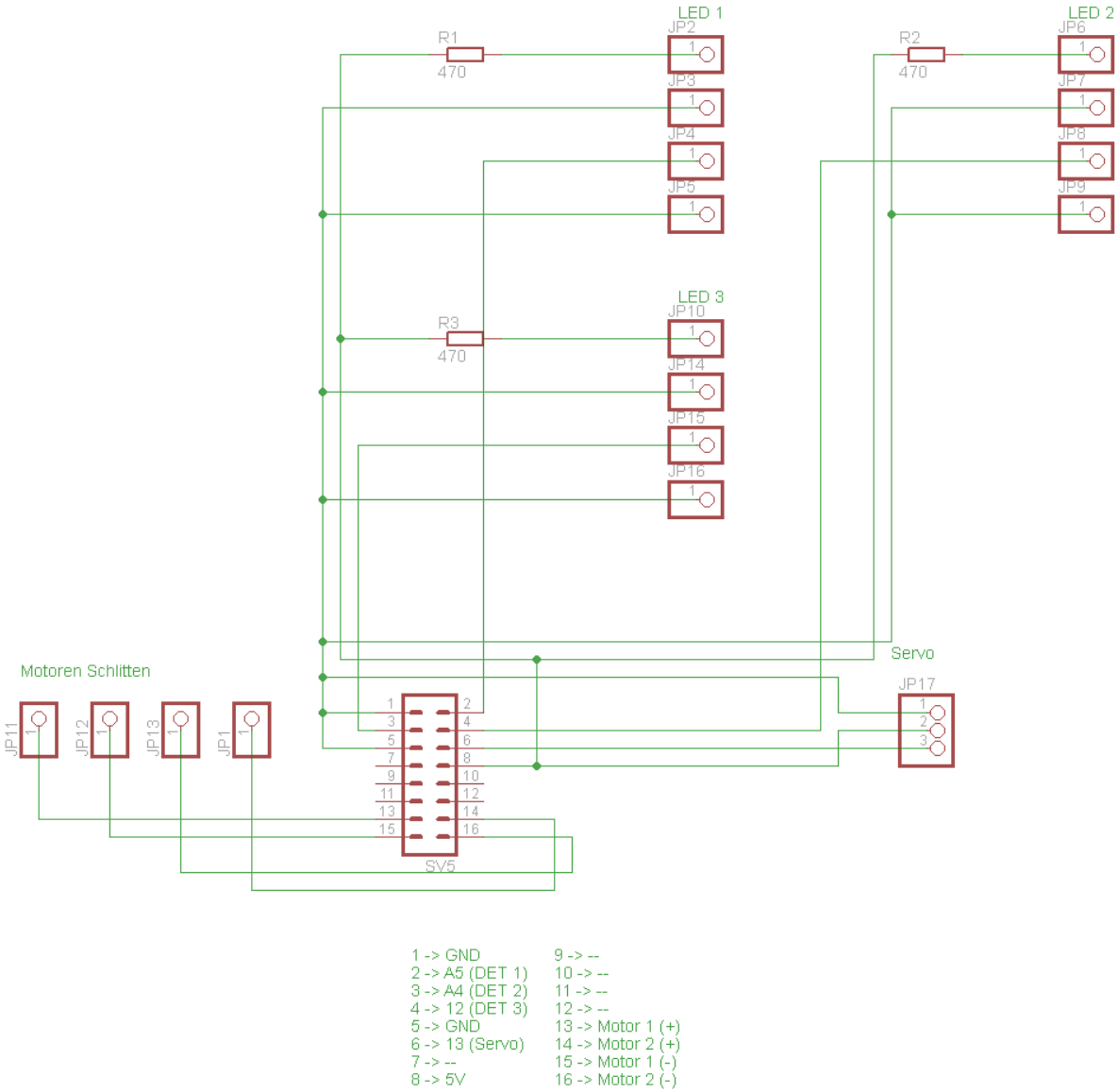


Abbildung 43: Schaltplan für die Elektronik am X-Achsenschlitten

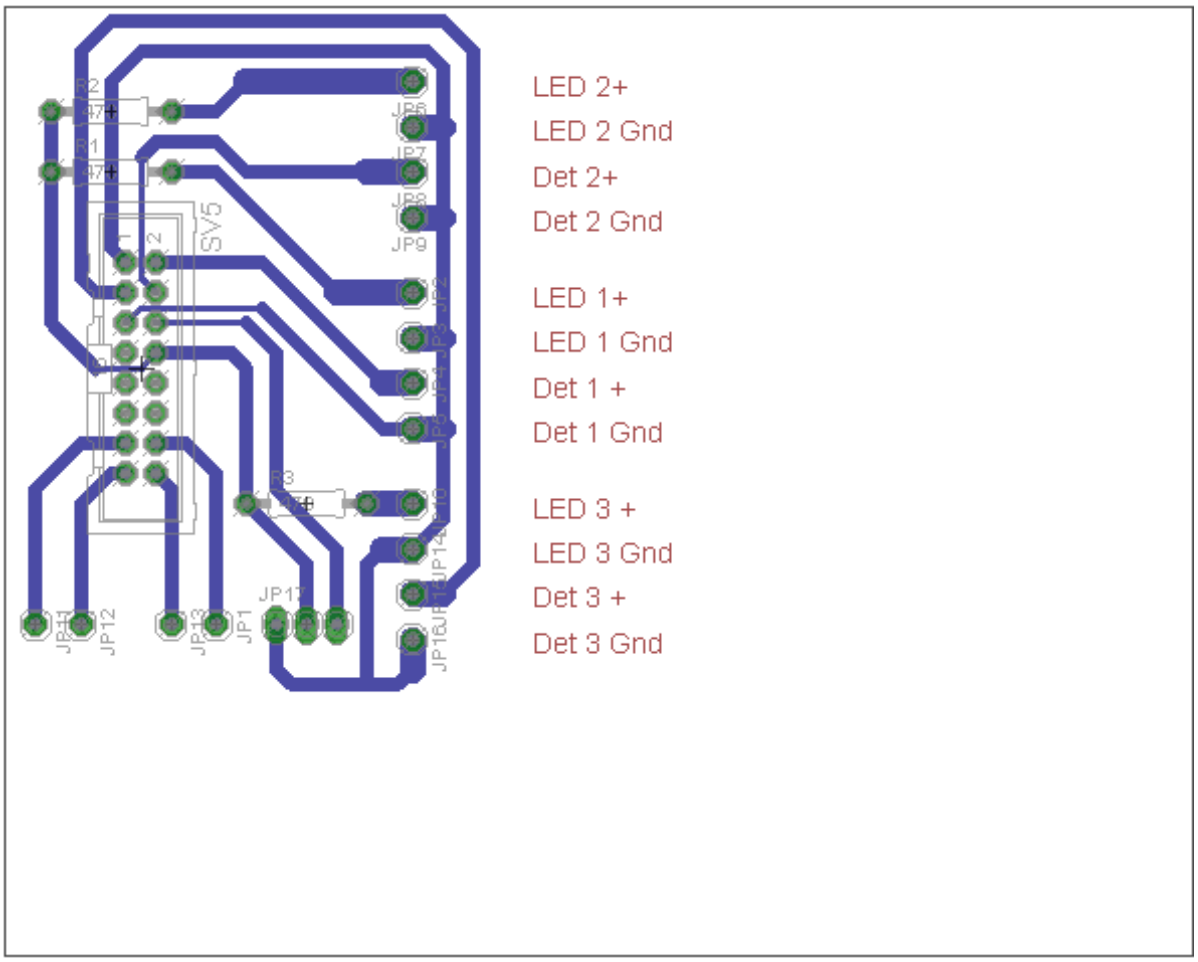


Abbildung 44: Platinenlayout für die Elektronik am X-Achsenschlitten

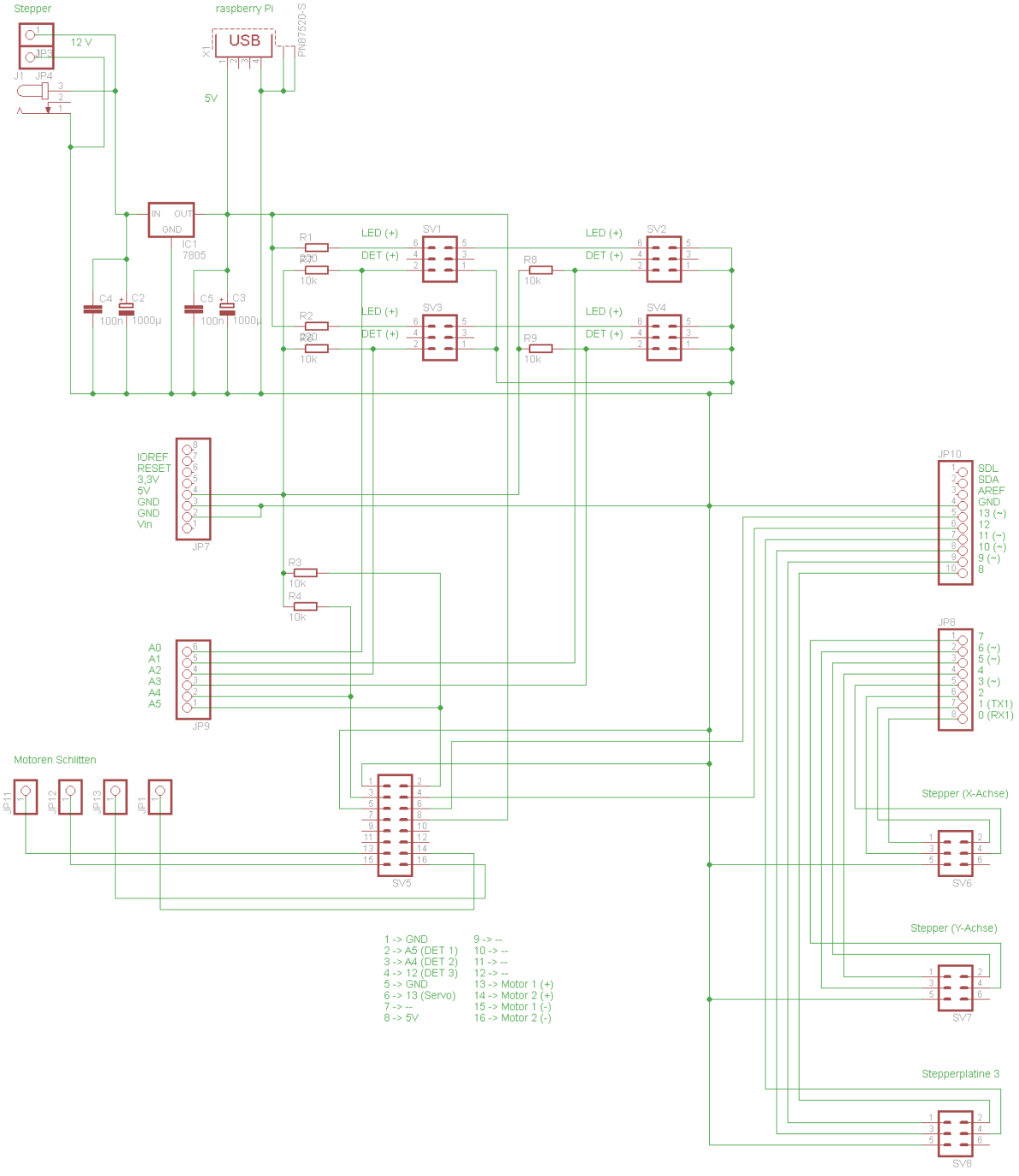


Abbildung 45: Schaltplan für die Hauptelektronik

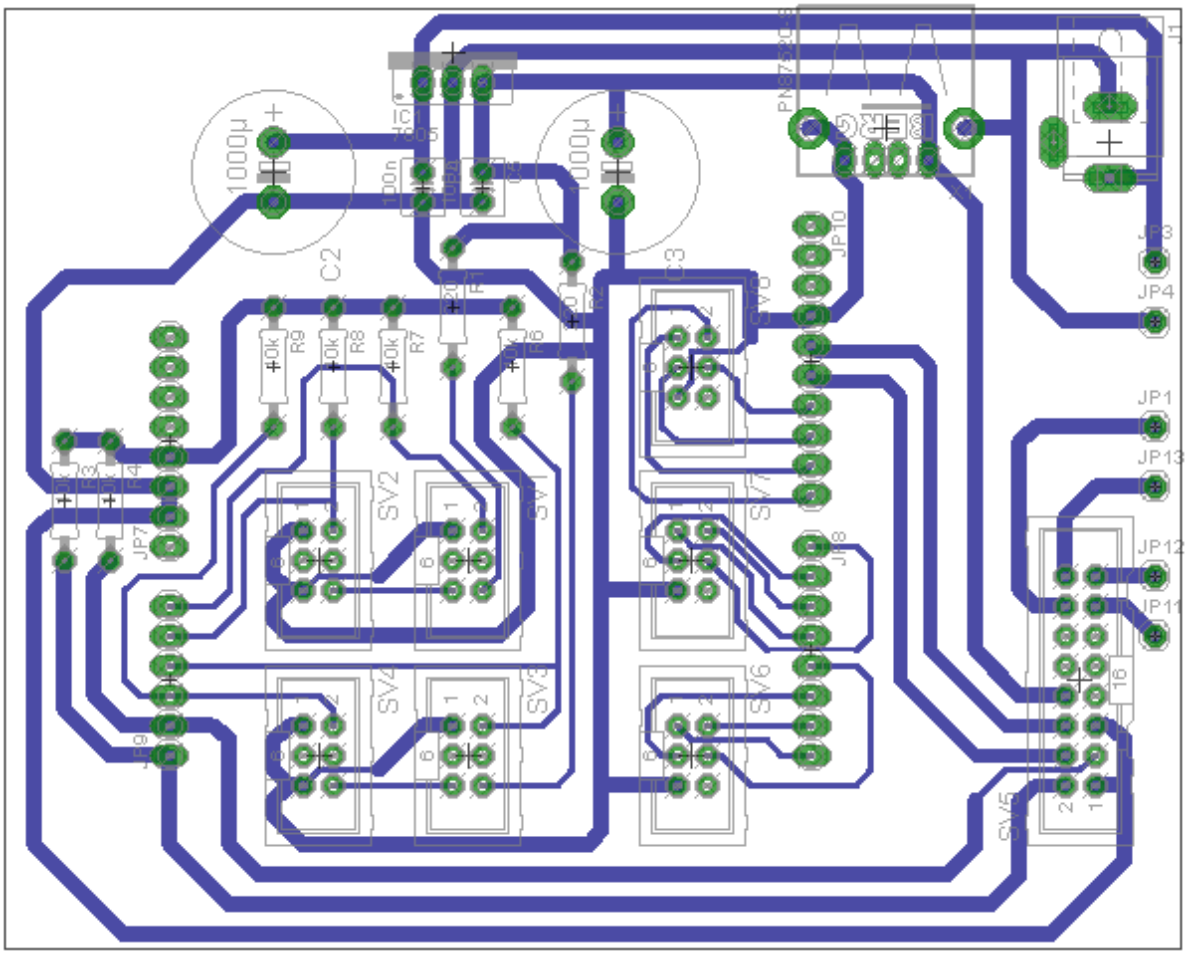


Abbildung 46: Platinenlayout für die Hauptelektronik

C. Dateiliste

Die folgende Liste beschreibt den Inhalt der beigelegten CD.

- **misc** - Beinhaltet Ordner für Konstruktionszeichnungen und die Testbilder.
  - **sketches** - Beinhaltet die Sketch Up-Dateien.
  - **design** - Beinhaltet die Konstruktionszeichnungen.
  - **test** - Beinhaltet die verwendeten Testbilder.
- **source** - Beinhaltet alle Quelldateien der BeadBox-Software.
  - **arduino** - Beinhaltet das Arduino-Projekt für die Hardwarekontrolle.
  - **bbconsole** - Beinhaltet das Projekt für die Konsolenanwendung.
  - **bbcontrol** - Beinhaltet das Projekt für das Hardwarekontrollprogramm.
  - **bbcore** - Beinhaltet das Projekt für die Hardwareschnittstellenschicht.
    - **common** - Beinhaltet die Klassen für die Konfigurationsdatei.
    - **controls** - Beinhaltet die Hardwarekontrollklassen.
    - **motors** - Beinhaltet Hilfsklassen für die Steuerung der Schrittmotoren.
    - **serial** - Beinhaltet Klassen für die Kommunikation über die serielle Schnittstelle.
  - **bbgui** - Beinhaltet das Projekt für die Benutzeroberfläche.
    - **controller** - Beinhaltet die GUI-Kontrollerklassen.
    - **items** - Beinhaltet die QtQuick-Items für die Vorschau und die Malfläche.
    - **models** - Beinhaltet die GUI-Modelklassen.
    - **qml** - Beinhaltet QML-Dateien für die GUI.
  - **bbprocessing** - Beinhaltet das Projekt für die Bildverarbeitungs- und Ablaufsteuerungsschicht.
    - **filters** - Beinhaltet die Klassen für die Bildfilter.
    - **plotter** - Beinhaltet die Klassen für die Ablaufsteuerung.
- **source-build** - Beinhaltet eine für den Raspberry Pi kompilierte Version der BeadBox-Anwendungssoftware inklusive Konfigurationsdatei.

Bezeichnung	Abmessung/Anmerkung	Stück	Einzelpreis	Gesamtpreis	Link
XY-Steuerung					
Präzisionswelle 10mm h6 geschliffen und gehärtet, 800mm	10 x 800mm	2	12,00 €	24,00 €	http://www.dold-mechatronik.de/Praezisionswelle-10mm-h6-geschliffen-und-gehaertet-800mm
Wellenhalter SH10	-	8	3,60 €	28,80 €	http://www.dold-mechatronik.de/Wellenhalter-SH10
Linearlager 10mm SCJ10UU	-	4	7,60 €	30,40 €	http://www.dold-mechatronik.de/Linearlager-10mm-SCJ10UU-Spiel-einstellbar
Linearlager 10mm SC10SUU	-	4	7,20 €	28,80 €	http://www.dold-mechatronik.de/Linearlager-10mm-SC10SUU
Trapezgewindespindel 12x6 P3 rechts 1 Meter	Tr 12x6	1	10,50 €	10,50 €	http://www.dold-mechatronik.de/Trapezgewindespindel-12x6-P3-rechts-1-Meter
Trapezgewindemutter 12x6 P3 R Kunststoff PA6.6 mit Aluminiumgehäuse	Tr 12x6	2	18,00 €	36,00 €	http://www.dold-mechatronik.de/Trapezgewindemutter-12x6-P3-rechts-Kunststoff-PA66-D26L24
Wellenkupplung RB flexibel D25L30 4,00/12,00mm		2	7,80 €	15,60 €	http://www.dold-mechatronik.de/Wellenkupplung-RB-flexibel-D25L30-400-1200mm
Vereinzelung und Magazin					
Aluminium Rundrohr	8mm/6mm, 1000mm	5	2,29 €	11,45 €	
PLEXIGLAS® XT (allround), Rohr, Farblos 0A070 GT	5mm/3mm, 200mm	0	1,39 €	- €	http://hbholzmaus.eshop.t-online.de/epages/Store7_Shop34800.sf/de_DE/?ObjectPath=/Shops/Shop34800/Products/%2200-5/3-1000%22
PLEXIGLAS® XT (allround), Rohr, Farblos 0A070 GT	40mm/34mm, 1000mm	0	9,15 €	- €	http://hbholzmaus.eshop.t-online.de/epages/Store7_Shop34800.sf/de_DE/?ObjectPath=/Shops/Shop34800/Products/%2200-38/32-1000%22
Bügelperlen					
Hama mini Perlen 2000 Stück	2,5 mm - schwarz	0	1,60 €	- €	http://www.hama.dk/perlen-7/mini-perlen-schwarz-2-5mm-501-18-idp1863/
Hama mini Perlen 2000 Stück	2,5 mm - weiß	0	1,60 €	- €	http://www.hama.dk/perlen-7/mini-perlen-weiss-2-5mm-501-01-idp1846/
Stiftplatte für mini Perlen - groß		0	2,80 €	- €	http://www.hama.dk/stiftplatten-6/mini-grosse-viereckige-stiftplatte-593-idp1296/
Hama midi Perlen 1000 Stück	207-03, Weiss	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-01-Perlen-wei%C3%9F-St%C3%BCck/dp/B0009JJKVM/
Hama midi Perlen 1000 Stück	207-03, Gelb	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-03-Perlen-gelb-St%C3%BCck/dp/B0009JJKXA/
Hama midi Perlen 1000 Stück	207-10, Grün	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-10-Perlen-grün-Stück/dp/B0009JJL1Q/
Hama midi Perlen 1000 Stück	207-17, Grau	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-Perlen-grau-1-000-Stück/dp/B0013GMWU6/
Hama midi Perlen 1000 Stück	207-18, Schwarz	1	1,40 €	1,40 €	http://www.amazon.de/Hama-Bügelperlen-schwarz-1000-207-18/dp/B0013GIMI2/
Hama midi Perlen 1000 Stück	207-04, Orange	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-04-Perlen-orange-Stück/dp/B0009JJKXU/
Hama midi Perlen 1000 Stück	207-11, Hellgrün	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-11-Perlen-hellgrün-Stück/dp/B0009JJL2K/
Hama midi Perlen 1000 Stück	207-20, Rotbraun	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-20-Perlen-rotbraun-Stück/dp/B0009JJL8E/
Hama midi Perlen 1000 Stück	207-05, Rot	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-05-Perlen-1000-Stück/dp/B0009JJKYE/
Hama midi Perlen 1000 Stück	207-09, Hellblau	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-09-Perlen-hellblau-Stück/dp/B0009JJL1G/
Hama midi Perlen 1000 Stück	207-06, Hellrot	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-06-Perlen-hellrosa-Stück/dp/B0009JJKZI/
Hama midi Perlen 1000 Stück	207-26, Hautfarbe	1	1,40 €	1,40 €	http://www.amazon.de/DAN-207-26-HAMA-Perlen-hautfarbe/dp/B000AYHN60/
Hama midi Perlen 1000 Stück	207-08, Blau	1	1,40 €	1,40 €	http://www.amazon.de/HAMA-207-08-Perlen-blau-Stück/dp/B0009JJL16/

Hama midi Perlen Stiftplatte - groß		4	1,45 €	5,80 €	http://www.amazon.de/Hama-234-HAMA-Multi-Platte/dp/B0009JJLRK/
Elektronik					
Arduino Leonardo Platine		1	24,95 €	24,95 €	http://www.conrad.de/ce/de/product/192458/Arduino-Leonardo-Platine-65163
Gabellichtschranke	TCST4103	10	0,30 €	3,00 €	http://www.pollin.de/shop/dt/OTYyOTc4OTk-/Bauelemente_Bauteile/Aktive_Bauelemente/Optoelektronik/Gabellichtschranke.html
Schrittmotorplatine-Bausatz	-	3	12,95 €	38,85 €	http://www.pollin.de/shop/dt/Mjc5OTgxOTk-/Bausaetze_Module/Bausaetze/Schrittmotorplatine_Bausatz.html
Schrittmotor PSM42BYGHW603	12V, 1,8°	2	15,95 €	31,90 €	http://www.pollin.de/shop/dt/NjQ1OTg2OTk-/Motoren/Schrittmotoren/Schrittmotor_PSM42BYGHW603_1_8_.html
Gleichstrom-Getriebemotor PGM-37DC12/21	-	1	7,95 €	7,95 €	http://www.pollin.de/shop/dt/NzE1OTg2OTk-/Motoren/DC_Getriebemotoren/Gleichstrom_Getriebemotor_PGM_37DC12_21.html
Gleichstrom-Getriebemotor CHM-2435-1	-	1	8,95 €	8,95 €	http://www.pollin.de/shop/dt/Nzc1OTg2OTk-/Motoren/DC_Getriebemotoren/Gleichstrom_Getriebemotor_CHM_2435_1.html
Wannenstecker	16-polig, gerade	4	0,12 €	0,48 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=22822
Wannenstecker	6-polig, gerade	10	0,17 €	1,70 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=85732
Pfostenbuchse, mit Zugentlastung	16-polig, gerade	4	0,14 €	0,56 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=14573
Pfostenbuchse, mit Zugentlastung	6-polig, gerade	10	0,24 €	2,40 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=53153
USB-Einbaubuchse	Typ A	2	0,22 €	0,44 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=22184
Hohlstecker-Buchse	Zentraleinbau, 2,1mm	2	0,23 €	0,46 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=8524
Elko radial	105°C, low ESR, RM 5,0mm	4	0,65 €	2,60 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=121272
Spannungsregler 1,5A positiv	TO-220	2	0,28 €	0,56 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=23443
Flachbandkabel AWG28	16-pol., farb., 3m-Ring	1	4,55 €	4,55 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=47670
Fotoplatine, Epoxyd, einseitig	150x100mm	2	1,90 €	3,80 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=7656
Steckernetzteil 1500A 12V		1	8,95	8,95 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=108296
Peripherie					
FAYTECH T7 SW Touchscreen		1	137,95 €	137,95 €	http://www.reichelt.de/index.html?ACTION=3;ARTICLE=87444
Sonstiges					
Aluminium U-Profil	8x10x200mm, silbereloxiert	1	5,49 €	5,49 €	
Metallschrauben	M5x10	2	1,69 €	3,38 €	
Metallschrauben	M5x20	2	2,29 €	4,58 €	
Kunstoff Protex Light	1000x500x3mm, schwarz	1	14,79 €	14,79 €	
Kunstoff Protex Light	500x500x3mm, schwarz	1	7,29 €	7,29 €	
Aluminium Winkel	10x10x2000mm	1	3,99 €	3,99 €	
Aluminium Winkel	24x24x1000mm	2	4,19 €	8,38 €	
Aluminium Quadratrohr	15x15x1000mm	4	3,39 €	13,56 €	
diverse M3 Schrauben und Muttern	-				
diverse M2 Schrauben und Muttern					

Quellenverzeichnis

[1] *Willkommen in einer bunten Welt der Perlen - HAMA*, http://www.hama.dk, Letzter Zugriff: 31.07.2013.

[2] *Raspberry Pi | An ARM GNU/Linux Box*, http://www.raspberrypi.org, Letzter Zugriff: 31.07.2013.

[3] *Arduino*, http://www.arduino.cc/, Letzter Zugriff: 31.07.2013.

[4] *Arduino Leonardo*, http://www.arduino.cc/en/Main/arduinoBoardLeonardo, Letzter Zugriff: 31.07.2013.

[5] *Arduino - Reference*, http://arduino.cc/en/Reference/HomePage, Letzter Zugriff: 11.08.2013.

[6] *Qt Project*, www.qt-project.org, Letzter Zugriff: 01.08.2013.

[7] *Qt Product*, http://qt.digia.com/product, Letzter Zugriff: 31.07.2013.

[8] *Cobra: High Speed Pick-and-Place - The Future in SMT Assembly*, http://www.essemtec.com/products.asp?ArtNr=Cobra, Letzter Zugriff: 30.08.2013.

[9] *Qt on Pi*, http://qt-project.org/wiki/qt-raspberrypi, Letzter Zugriff: 03.08.2013.

[10] *Native Build of Qt 5 on a Raspberry Pi*, http://qt-project.org/wiki/Native_Build_of_Qt5_on_a_Raspberry_Pi, Letzter Zugriff: 08.08.2013.

[11] *Beginner's guide to cross-compile Qt5 on RaspberryPi*, http://qt-project.org/wiki/RaspberryPi_Beginners_guide, Letzter Zugriff: 03.08.2013.

[12] *Schrittmotoren*, http://www.rn-wissen.de/index.php/Schrittmotoren, Letzter Zugriff: 09.08.2013.

[13] *Gabellichtschranke*, http://www.rn-wissen.de/index.php/Gabellichtschranke, Letzter Zugriff: 16.08.2013.

[14] *Pullup Pulldown Widerstand*, www.rn-wissen.de/index.php/Pullup_Pulldown_Widerstand, Letzter Zugriff: 16.08.2013.

[15] *Platinenherstellung mit der „Foto-Transfer-Technik“*, http://www.rn-wissen.de/index.php/Platinenherstellung_mit_der_%22Foto-Transfer-Technik%22, Letzter Zugriff: 16.08.2013.

[16] *Pulsweitenmodulation*, http://www.rn-wissen.de/index.php/Pulsweitenmodulation, Letzter Zugriff: 11.08.2013.

[17] *Vibrationswendelförderer*, http://www.deprag.com/schraubtechnik/produkte/zufuehrtechnik/zufuehrtechniken/vibrationswendelfoerderer.html, Letzter Zugriff: 09.08.2013.

[18] *Inversion of Control Containers and the Dependency Injection pattern*, http://www.martinfowler.com/articles/injection.html, Letzter Zugriff: 11.08.2013.

[19] *Qt Serial Port*, http://qt-project.org/doc/qt-5.1/qtserialport/qtserialport-index.html, Letzter Zugriff: 01.09.2013.

[20] *ATmega16U4/32U4 Preliminary*, http://www.atmel.com/Images/doc7766.pdf, Letzter Zugriff: 04.09.2013.

[21] Meltwater, *Pi Setup*, The MagPi - Issue 2, Juni 2012, S. 4.

[22] Meltwater, *SD Card Setup*, The MagPi - Issue 2, Juni 2012, S. 6.

[23] Carsten Meyer, *Portalfäse selbst gebaut*, c't Hardware Hacks, Januar 2013, S. 38.

[24] Robert W. Floyd und Louis Steinberg, *An adaptive algorithm for spatial grey scale*, Proceedings of the Society of Information Display 17, 1976, S. 75–77.

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 11. September 2013

Henning Ziegler