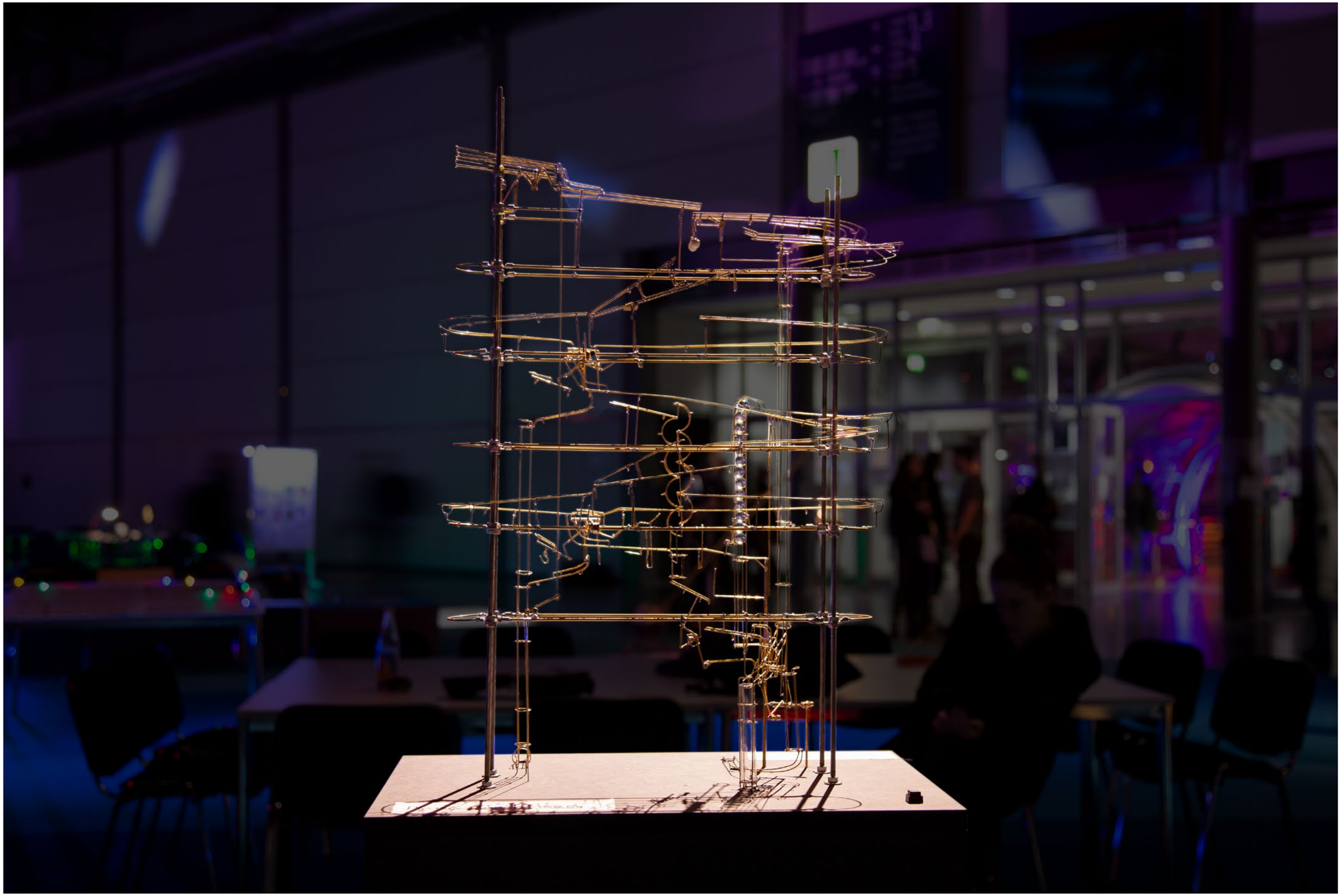
But does it run Linux?

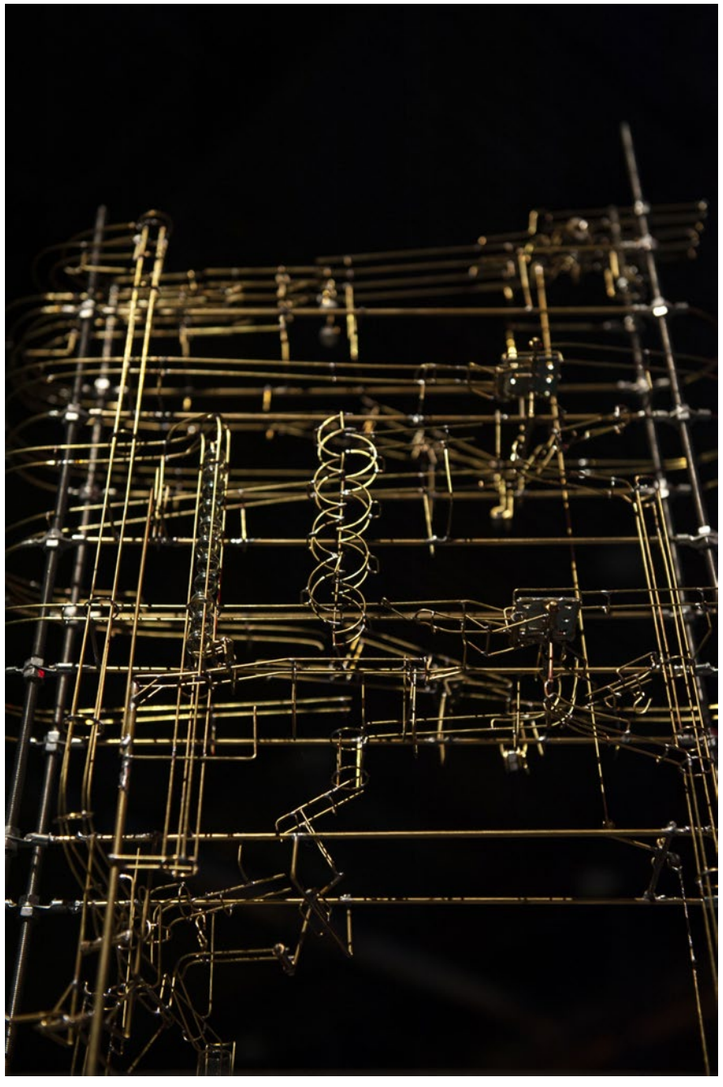
a clocked binary marble calculator

Prokjektdokumentation
Franziska Schneider

Stand: 10.01.2019



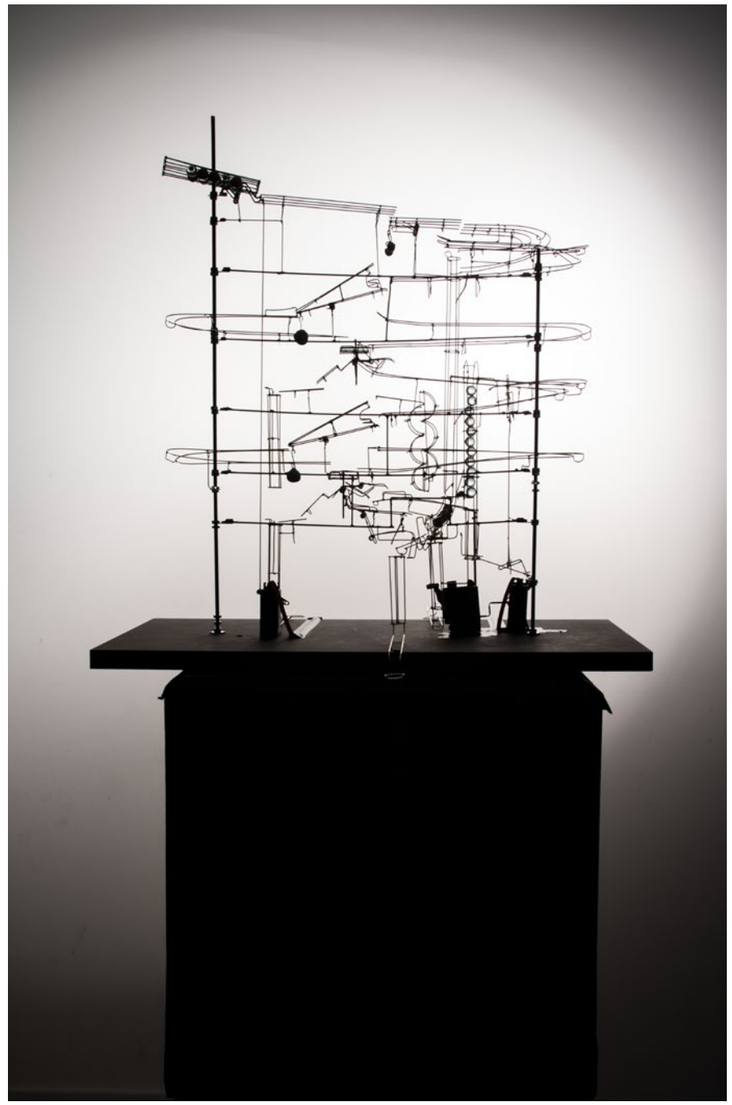
‚But does it run Linux?‘ auf dem 35C3



Inhalt

Kurze Versionsgeschichte

Das Konzept und die erste Version von *But does it run Linux?* entstanden im Wintersemester 2017/18 als Semesterarbeit im Rahmen des Designprojekts *(Un)control* unter der Leitung von Prof. Andreas Muxel an der Fakultät für Gestaltung der Hochschule Augsburg. Nachdem der Prototyp nach einem Abend des Betriebs durch allgemeine Materialermüdung ein frühzeitiges Ende fand, entstand Version 2 des Rechenwerks im Herbst 2018 für den 35. Chaos Communication Congress. Diese Dokumentation thematisiert zum Großteil die Entwicklung und den Bau des Prototypen. Auf den Neubau wird in einem eigenen Abschnitt gesondert eingegangen.



Der Protoyp des Rechenwerks

Alltägliche Algorithmen?

Computer sind Universalmaschinen. Mit den Informationen, die wir ihnen geben, können sie tun, was immer wir ihnen vorgeben. Was für uns jedoch Informationen sind, sind für einen Computer nur Zahlen: nicht endende Ketten von Einsen und Nullen, deren Bedeutung sich erst aus unserem Gebrauch von ihnen ergibt.
Die eigentliche Funktion eines Computers liegt darin, diese Zahlen zu verarbeiten und mathematische und logische Operationen mit ihnen durchzuführen. Die grundlegenste aller Operationen ist dabei die Addition, auf die alle anderen Rechenarten zurückgehen. Vom Smartphone in der Hosentasche bis zur Steuerungselektronik in unseren Autos durchdringen Computer heute unseren gesamten Alltag. Sobald die Geräte in Betrieb sind, werden jeder Sekunde um uns herum ganz ohne unser Zutun Millionen von Berechnungen gemacht. Das macht die simple Addition zweier Zahlen zu einem der wichtigsten Algorithmen unseres Alltags.

Das Konzept

We switch them on, they do stuff for us. Is it magic? What do our computers do? This brand new mechanical 0.1Hz calculator featuring 1-bit architecture and 16bit of RAM is constructed to answer that question by bringing elemental computer logic to a human scale. It does not run Linux, but marbles. See them rolling. Infotext der Postkarte zur Ausstellung am 18.01.2018

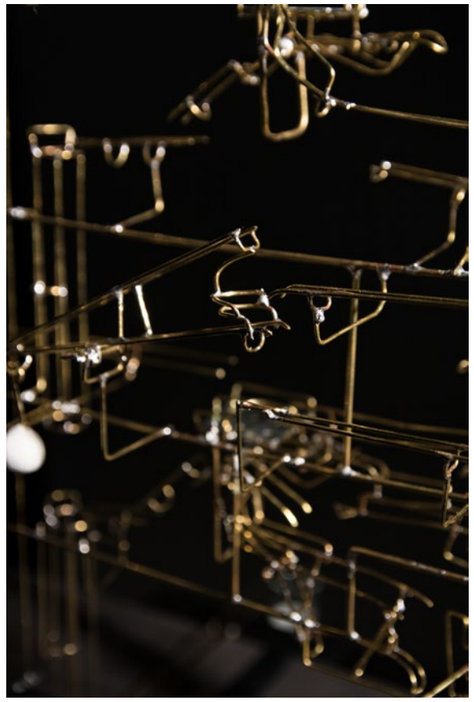
Maschinelles Denken prägt und beeinflusst unser tägliches Leben maßgeblich – und dennoch bleiben die Prozesse im I nneren der Maschine für uns unfassbar.

Ein getaktet arbeitendes, mechanisches Addierwerk, das zwei binäre Zahlen addieren kann, überträgt einige dieser verborgenen Abläufe räumlich und zeitlich in menschliche Dimensionen. Die mechanische Umsetzung einer elektronischen Schaltung macht aus einem starren logischen Prinzip ein fragiles, kleinsten Umwelteinflüssen unterworfenes System.

Wie unterscheidet sich das Konzept des Addierwerks von existierenden Binärrechnern?

Eine Besonderheit des seriellen Addierwerkes ist die Handhabung der Zahlenein- und -ausgabe. Während die binäre Information in vielen existierenden mechanischen Binärrechnern sehr abstrakt durch beispielsweise Schalterpositionen oder sich drehende Zahnräder codiert ist, liegen die Zahlen im seriellen Murmeladdierwerk als sichtbare Abfolge von optisch unterscheidbaren Kugeln vor. Sie dienen also nicht nur der Kraftübertragung durch ihre Masse, sondern sind selbst die physische Repräsentation von Bits.
In Reagenzgläsern „gespeicherte“ Bitfolgen, die einfach in die Maschine geladen werden können, erlauben es, das Rechenwerk auch ohne Kenntnis des binären Systems zu bedienen.
Eine weitere Besonderheit des seriellen Addierwerks ist seine getaktete, zyklische Arbeitsweise. Diese veranschaulicht einen wichtigen Aspekt der Funktion eines Computers, der von parallel arbeitenden Addierwerken nicht gezeigt wird und erlaubt es, theoretisch unendlich viele Berechnungen ohne Unterbrechung hintereinander durchzuführen, da sich das Rechenwerk nach jedem Takt in seinen Ausgangszustand zurückversetzt.

Seitenansicht des Prototypen



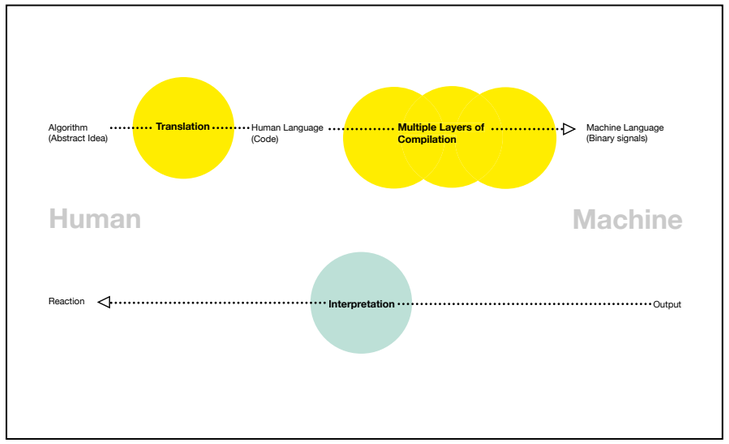
Detailansicht eines Halbaddierer-Bauelements

Zum Titel

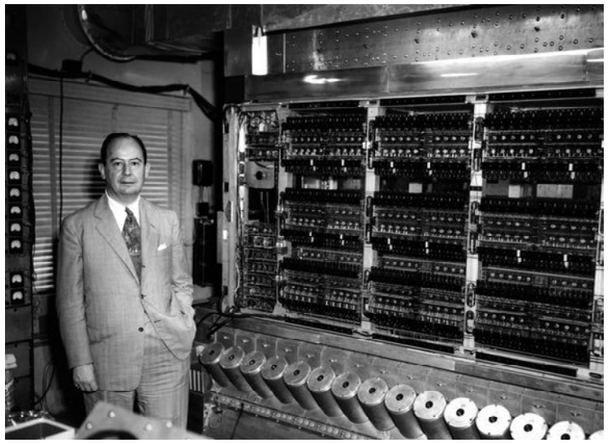
Der Titel „But does it run Linux?“ ist eine Anspielung auf einen Scherz, der häufiger im Internet zu finden ist:
Unter Veröffentlichungen von selbstgebauten einfachen Rechnern fand sich meistens mindestens ein Kommentar, in dem ironisch danach gefragt wurde, ob das jeweilige Gerät auch in der Lage sei, diverse anspruchsvolle Software zu betreiben.
„But does it run Linux?“ ist dabei der zeitlose Klassiker.

Ideenfindung

Die Brücke zwischen der analogen und der digitalen Welt und die Nichtwahrnehmbarkeit von digitalen Prozessen war der Ausgangspunkt der Ideenfindung.



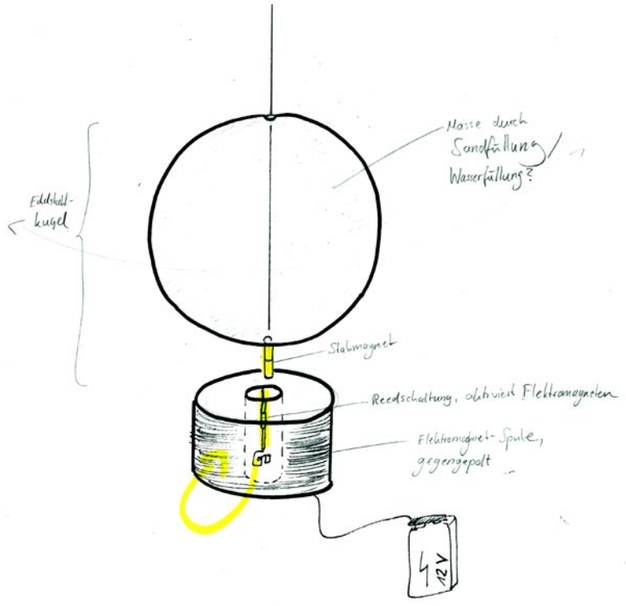
Überlegungen zur Übersetzung zwischen Mensch und Maschine



Mathematiker John von Neumann vor dem MANIAC I
Quelle: Dyson, George: Turings Kathedrale, S.280

Zu Beginn lag der Fokus vor allem auf der Übersetzung von menschlichen Ideen in Maschinencode.
Die Recherche zu Computern der ersten Generation, die komplett ohne höhere Programmiersprachen programmiert wurden und insbesondere der Rechner MANIAC I (Bild rechts), dessen Rechenzyklen tatsächlich von außen beobachtet werden konnten, führten schließlich zu dem Einfall, die unsichtbaren Vorgänge in der Maschine sichtbar zu machen.

6m lang und 2kg schwer: das ursprünglich als Taktgeber vorgesehene Pendel aus Stahl

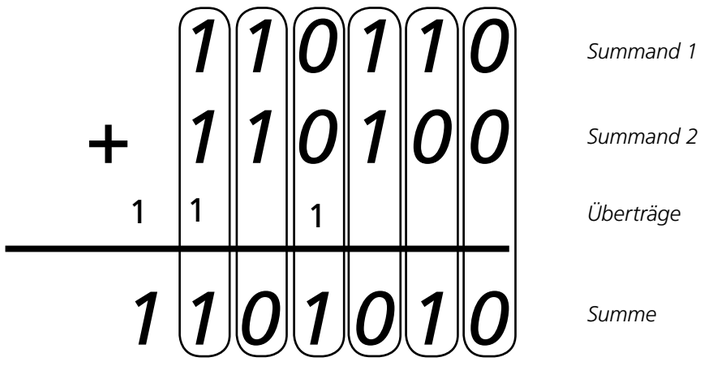


Skizze eines „künstlichen“ Foucaultschen Pendels

Noch vor der Idee zum Bau eines mechanischen Addierwerks stand der Gedanke, einen Computer nicht wie üblich mit einer nicht wahrnehmbaren, möglichst hohen Taktfrequenz zu betreiben, sondern durch einen physikalischen, langsamen und durch Umwelteinflüsse fehleranfälligen Taktgeber, der von Menschen erfasst werden kann.
Daraus entwickelte sich in der Folge die Idee, ein komplett mechanisches, von einem Foucaultschen Pendel angetriebenes binäres Rechenwerk zu bauen. Die mechanische binäre Addition wurde sehr bald zum interessanteren Aspekt des Projektes, weswegen das Pendel letztendlich einer unkomplizierteren elektronischen Steuerung wich und der Fokus damit voll auf der Darstellung der binären Addition lag.

Das Prinzip der binären Addition

Die Addition zweier binärer Zahlen, wie sie von Computern durchgeführt wird, unterscheidet sich im Grunde nicht von der dezimalen Addition. Zwei Zahlen werden addiert, indem ihre Stellen einzeln berechnet werden:



Durch die Beschränkung auf die Ziffern 0 und 1 kann die Summe an einer Stelle ebenfalls nur den Wert 1 oder 0 haben. Wird die Summe größer als 1, so entsteht ein Übertrag auf die nächste Stelle. Dadurch, dass bei der Berechnung einer Stelle jedes Mal ein eventueller Übertrag berücksichtigt werden muss, besteht die Addition eigentlich aus zwei Teilberechnungen:

Summand 1 + Summand 2 = Zwischensumme

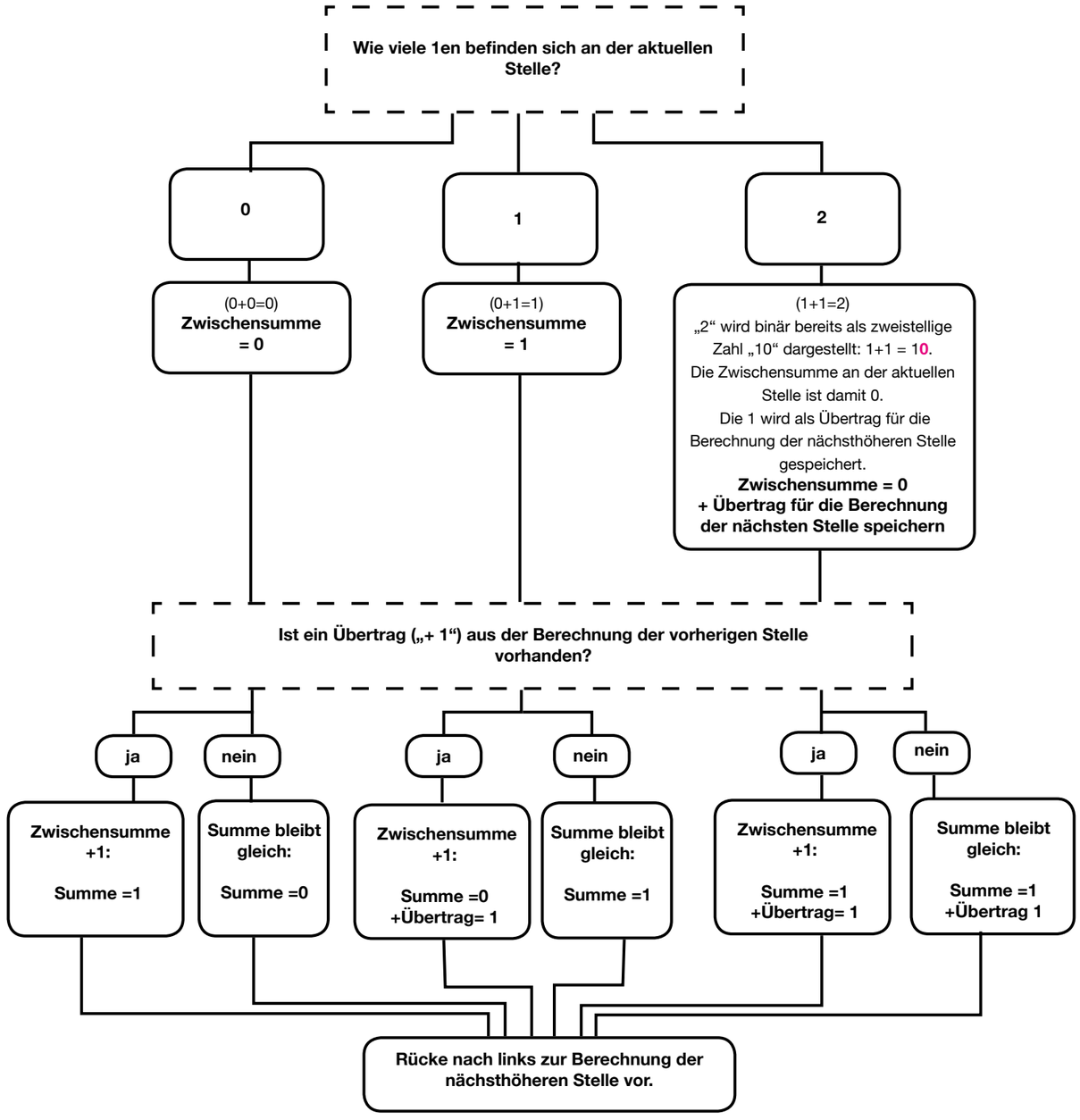
Zwischensumme + Übertrag = Summe

Die Anzahl der möglichen Summen einer Teiladdition ist durch die Austauschbarkeit der Summanden auf 3 begrenzt und kann daher einfach in einer Tabelle dargestellt werden (vgl. Tabelle rechts). Binäre Addition kann also auf eine Auswahl eines von mehreren bedingten Entscheidungspfaden reduziert werden (vgl. dazu das Additions-Flowchart auf der nächsten Seite). Das mechanische Addierwerk ist letztendlich die reale Umsetzung dieser Entscheidungspfade. Serielle Addierwerke eignen sich besonders zur Veranschaulichung diese Prinzips, da sie genau wie bei der Schriftlichen Addition von Hand die Zahlen stelle für Stelle von der niedrigsten zur Höchsten abarbeiten.

Summand 1	Summand 2	Summe
1	0	1
1	1	0 + Übertrag
0	0	0
0	1	1

Mögliche Ergebnisse einer Teiladdition

Die Logik zur Durchführung einer der beiden Teilberechnungen wird *Halbaddierer* genannt. In Computern wird diese durch logische UND- und exklusive ODER-Gatter realisiert.
Zwei hintereinander geschaltene Halbaddierer können so verwendet werden, um eine Vollständige Addition durchzuführen. Bei elektronischen Addierern ist zusätzlich ein ODER-Gatter zur Signalzusammenführung notwendig, welches sich später in der mechanischen Ausführung jedoch als überflüssig erwies.

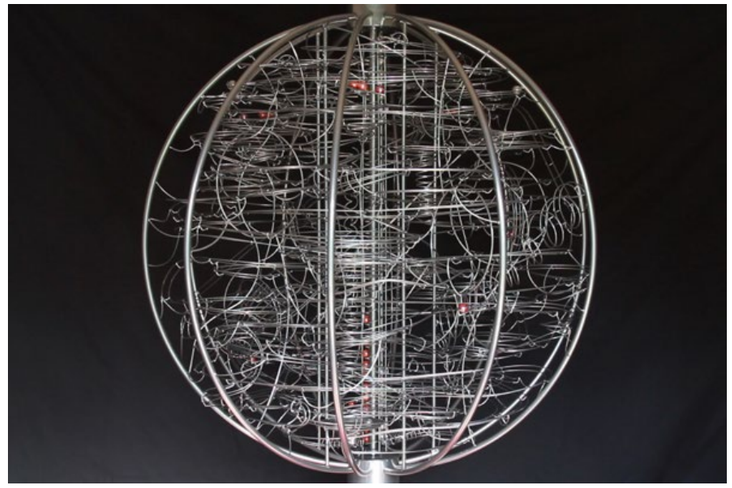


Flowchart: Binäre Addition als Reihe bedingter Entscheidungen

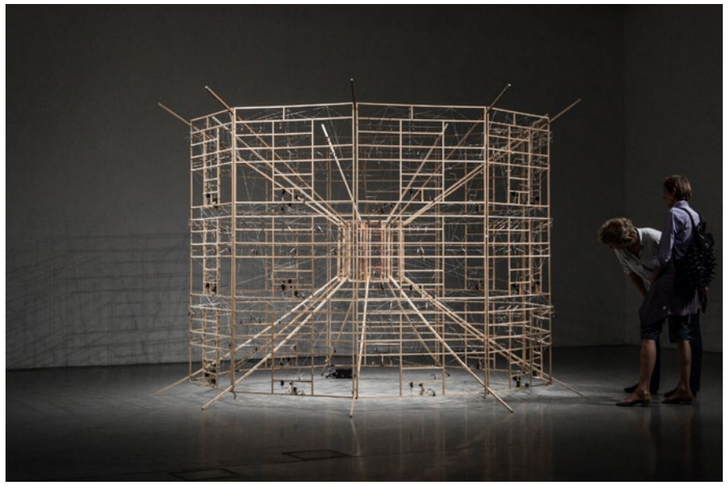
Recherche

Ausgangspunkt der Recherche waren bereits existierende mechanische Computer und Addierer. Vor allem relevant waren Umsetzungen der Maschinenlogik mit ungewöhnlichen Medien und verschiedene Arten und Weisen, binäre Information darzustellen. Inspiration lieferten unter anderem ein Domino-Rechner von Matt Parker, der binäre Holzcomputer „Digi-Comp II“, die Installation „Rechnender Raum“ von Ralf Becker sowie der gleichnamige Aufsatz von Konrad Zuse, die Szene der Bauer sogenannter „Minecraft-Computer“ und viele weitere im Internet veröffentlichte binäre und dezimale mechanische Rechner.
Später lieferten kinetische Murmelskultpuren, dabei insebsondere die des australischen Künstlers David Morell, und unzählige auf Youtube zu findende Videos von Murmelbahnen aus verschiedenen Materialien Inspiration für die mechanische Umsetzung der Schaltlogik.

4-Bit Domino-Computer des Mathematikers Matt Parker, 2012
Quelle: https://www.scientificamerican.com

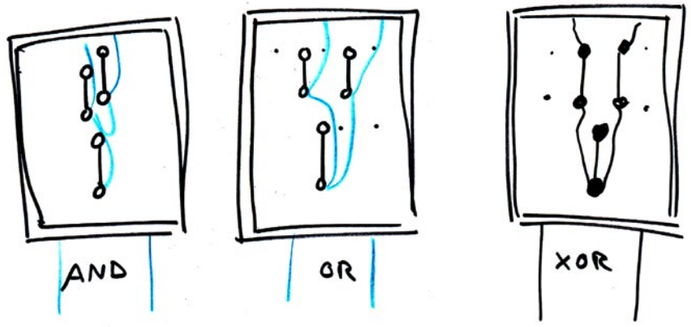


Murmelskulptur „#82“ des autralischen Künstlers David Morell
Quelle: http://www.rollingballsculpture.com.au/rolling_ball_sculpture

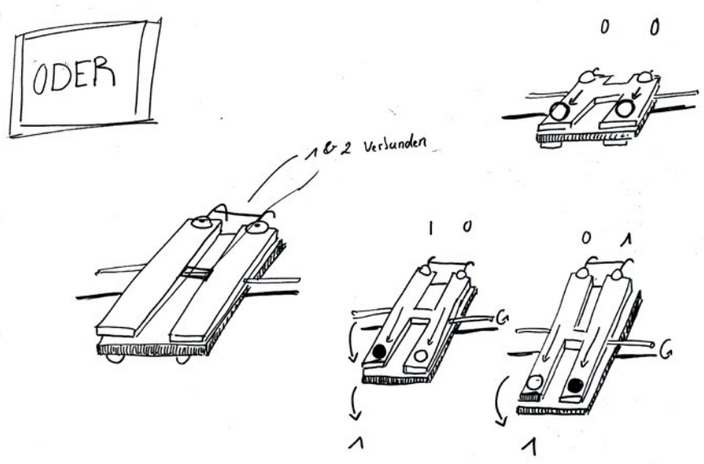


Installation „Rechnender Raum“ von Ralf Becker, 2007
Quelle: http://www.rlfbckr.org/work/rechnender-raum

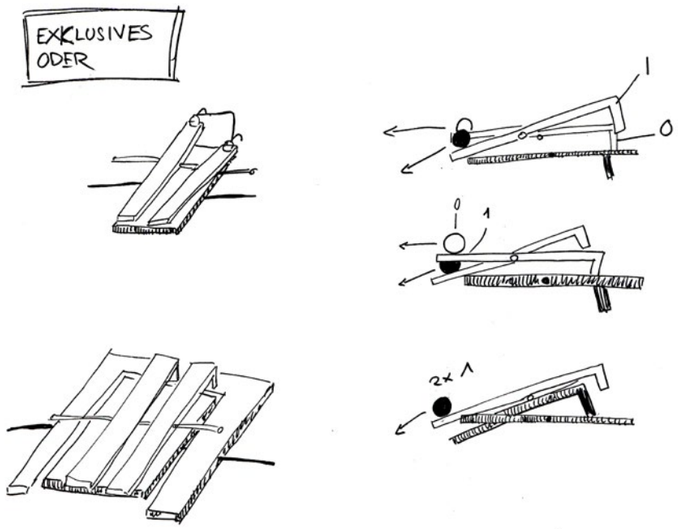
Digicomp II (Nachbau)
Quelle: evilmadscientist.com



Erste Ideen zur mechanischen Umsetzung der Gatterlogiken per Fadenzug



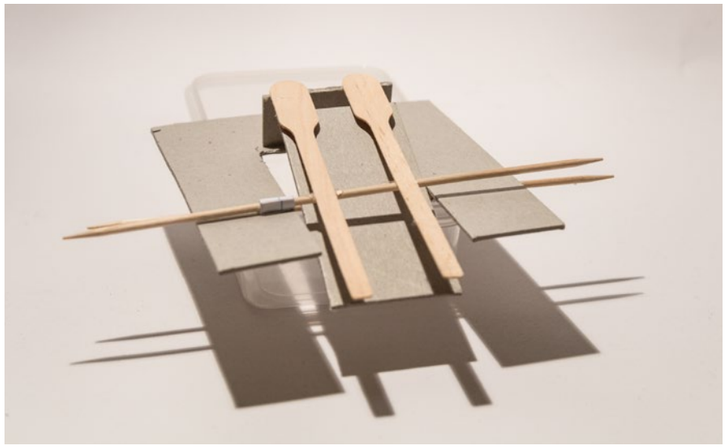
Schematische Darstellung der Realisierung der ODER-Logik über Hebel

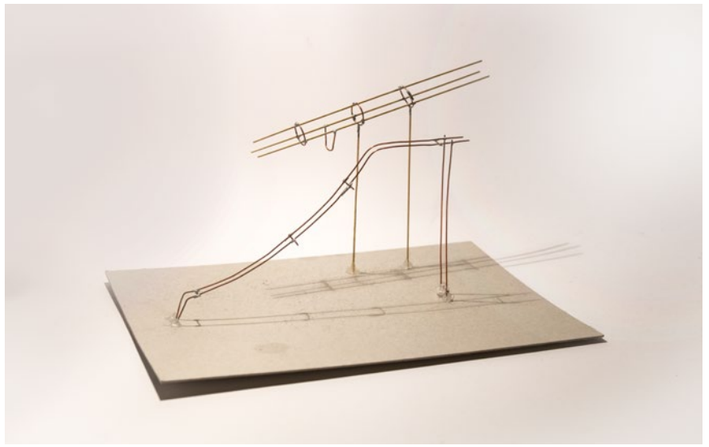


Skizze zur mechanischen Umsetzung der exklusiven ODER-Logik

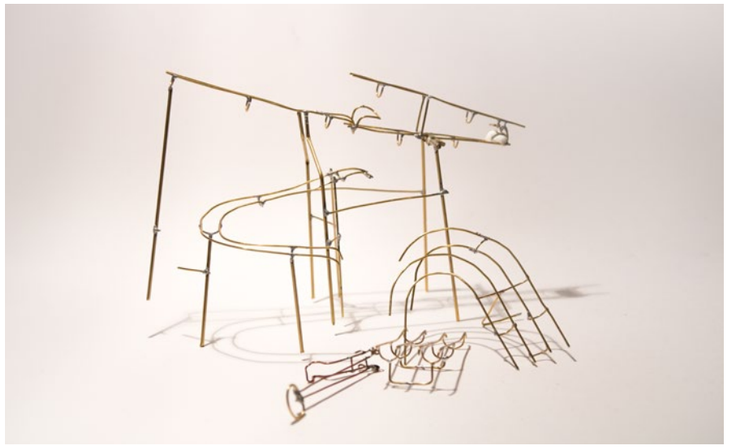
Die Entwicklung der Umsetzung

Die Entwicklung der mechanischen Umsetzung nahm neben der tatsächlichen Bauphase der größten Teil der Projektzeit ein und stellte die größte Herausforderung dar.
Erste Versuche bestanden darin, die Logik der Gatter entsprechend des Schaltplanes für einen elektronischen Addierer zuerst über Fäden und später über von verschieden schweren Murmeln betätigte Hebel umzusetzen (siehe Skizzen oben). Während die Hebel-Mechanik wenig praktikabel war, ist die Codierung der Bits in zwei durch ihr Gewicht unterscheidbaren Murmelsorten erhalten geblieben.

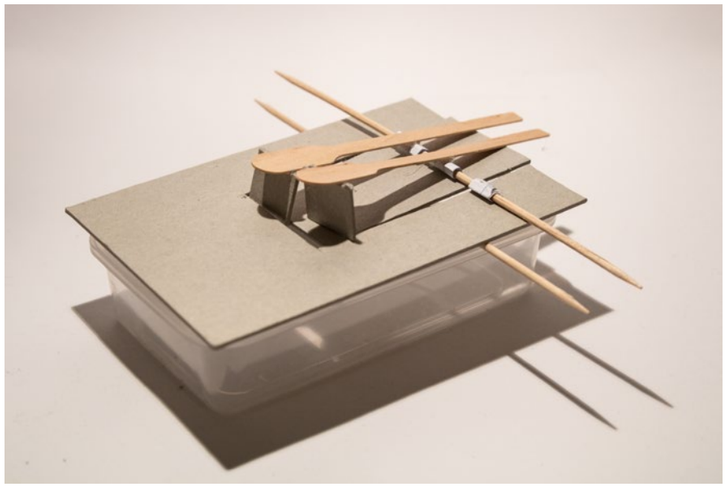




Erste Versuche mit Kupferdraht

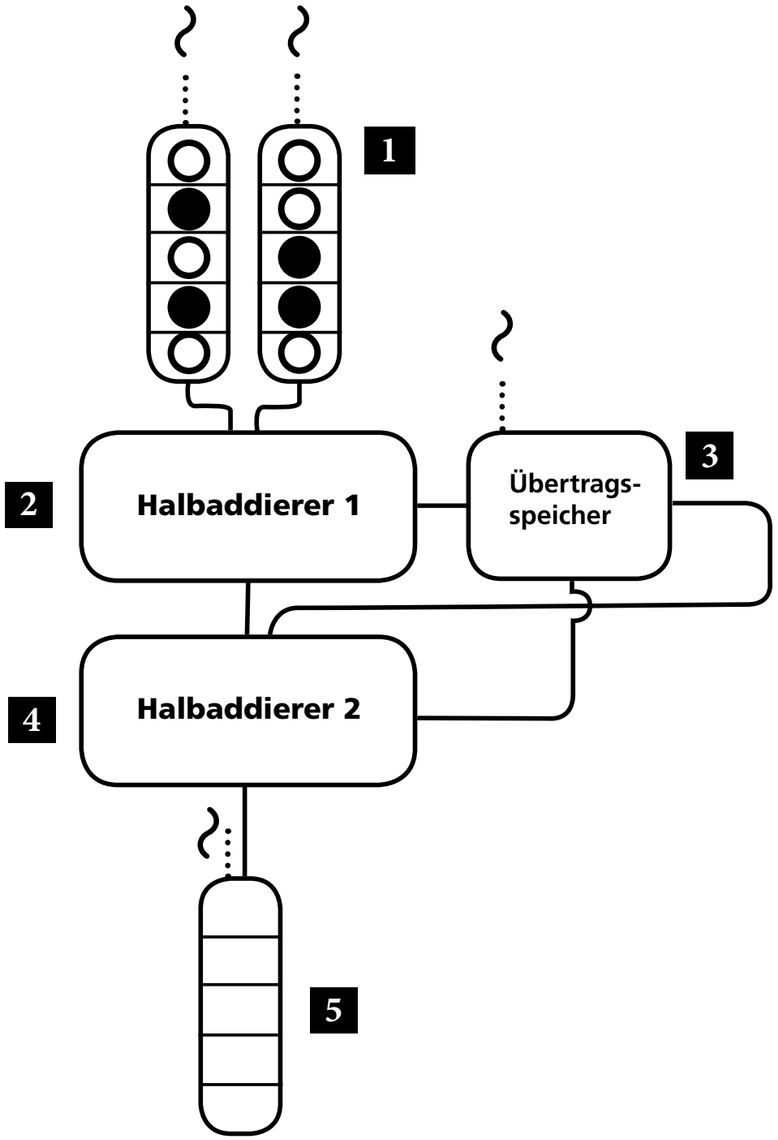


Prototypen einiger Bauelemente aus Kupfer- und Messingdraht



Prototypen der Hebel-Gatter aus Pappe

Nachdem die Codierung der Information in Form von Murmeln festgelegt war, bot sich die Realisierung des Addierers als Kugelbahn an.
Nach einigen Versuchen mit Kupferdraht fiel die Wahl des Baumaterials schließlich auf Messingdraht. Für diese Entscheidung sprachen die vielseitige Einsetzbarkeit und die Ästhetik des Materials sowie die Möglichkeit, es mit gängigem Werkzeug verarbeiten zu können. Für höhere Stabilität ist der Messingdraht in sich verdrillt.



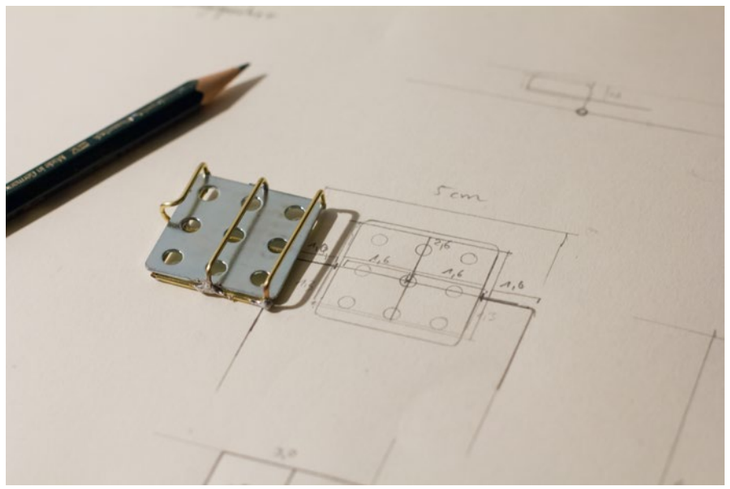
Funktionsweise

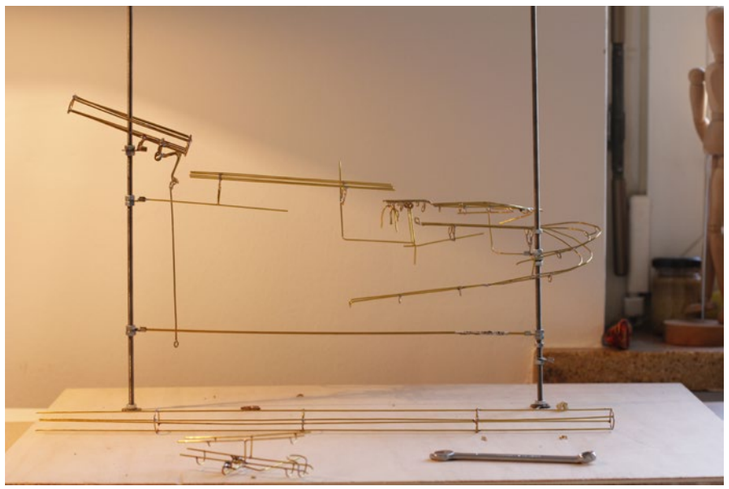
Die Mechanik des Addierwerks folgt mit geringen Anpassungen dem logischen Aufbau eines seriellen Addierwerks. Somit ist es aus zwei Input-Schieberegistern (1), zwei Halbaddierern (2,4), einem Speicher für das Übertragsbit (3) und einem Output-Register aufgebaut. Die Register und der Übertragsspeicher sind an das Taktsignal (~) angeschlossen. Ein Arbeitstakt läuft wie folgt ab:

Die Input-Register enthalten die beiden zu addierenden Zahlen als Bitfolg mit der niedrigsten Stelle voraus. Zu Beginn jedes Takts werden die gespeicherten Bits um eine Stelle nach vorne verschoben und die vordersten Bits somit aus dem Speicher geschoben.
Diese werden im Halbaddierer 1 miteinander verrechnet. Von den möglichen Ergebnissen,"*null*", *„eins, kein Übertrag "* oder *„Übertrag"* werden entsprechend an den Übertragsspeicher und den Halbaddierer 2 weitergegeben.

Parallel dazu wird der Wert, der im vorherigen Takt im Übertrag gespeichert wurde, zum Taktbeginn in den Halbaddierer 2 geladen und anschließend zum Ergebnis aus der ersten Berechnung addiert. Die Summe, die auch hier *„null"*, *„eins, kein Übertrag"* oder *„null, mit Übertrag"* betragen kann, wird in das Output-Register geschrieben bzw. an den Übertragsspeicher übergeben.

Der gesamte Prozess wird so lange wiederholt, bis sowohl die Input-Register als auch der Übertragsspeicher „geleert" sind bzw. den Wert „null" besitzen. Die Anzahl der für die komplette Addition ist somit die Länge der zu addierenden Bitfolgen +1.





Der Bauprozess

Die einzelnen Bauteile entstanden hauptsächlich nach dem Try-and Error-Prinzip. Viele Elemente habe ich Schritt für Schritt angepasst oder mehrmals komplett neugebaut. Da sich das Gewicht der schwereren Murmeln mehrmals änderte, mussten auch die Bauteile mehrmals umgebaut oder verändert werden. In einzelnen Fällen wie der Verteilerplattform (mittleres Bild) der Halbaddiererlogik existierte im Vorfeld eine genaue Konstruktionszeichnung.
Die Notwendigkeit bestimmter Bauteile wie z.B. eines Buffers ergab sich oft erst während des Bauprozesses.

Die komplette Murmelbahn entstand stückweise von oben nach unten. Die Befestigung der einzelnen Elemente mit Kabelschuhen an zwei Gewinden erlaubte es bis zum Schluss, einzelne Elemente oder die gesamte Mechanik in ihrer Position anzupassen. Die Sperrholzplatte wurde nach der ersten Bahnebene durch eine schwarze MDF-Platte ersetzt. Sämtliche Verbindungsstellen sind weich gelötet.

Das Rechenwerk

Die finale Umsetzung der Murmelbahn ist rund 60cm hoch und 50cm breit. Es wurden rund 10m Messingdraht verbaut. Zwei Gewinde und sechs Querverstrebungen verbinden die 34 einzelnen verlöteten Bauteile. Im Folgenden werden die Elemente der Mechanik im Detail vorgestellt.

Anmerkung: Fotos und Beschreibungen im folgenden beziehen sich jeweils auf den Prototypen des Modells, der jedoch dieselbe Funktionalität wie die neugebaute Version besitzt. Auf Unterschiede zwischen beiden Versionen wird im nachfolgenden Abschnitt „Neubau des Rechenwerks“ eingegangen.

Ein Video des Prototypen in Betrieb ist zu finden unter:
https://vimeo.com/253022967

Die Mechanik

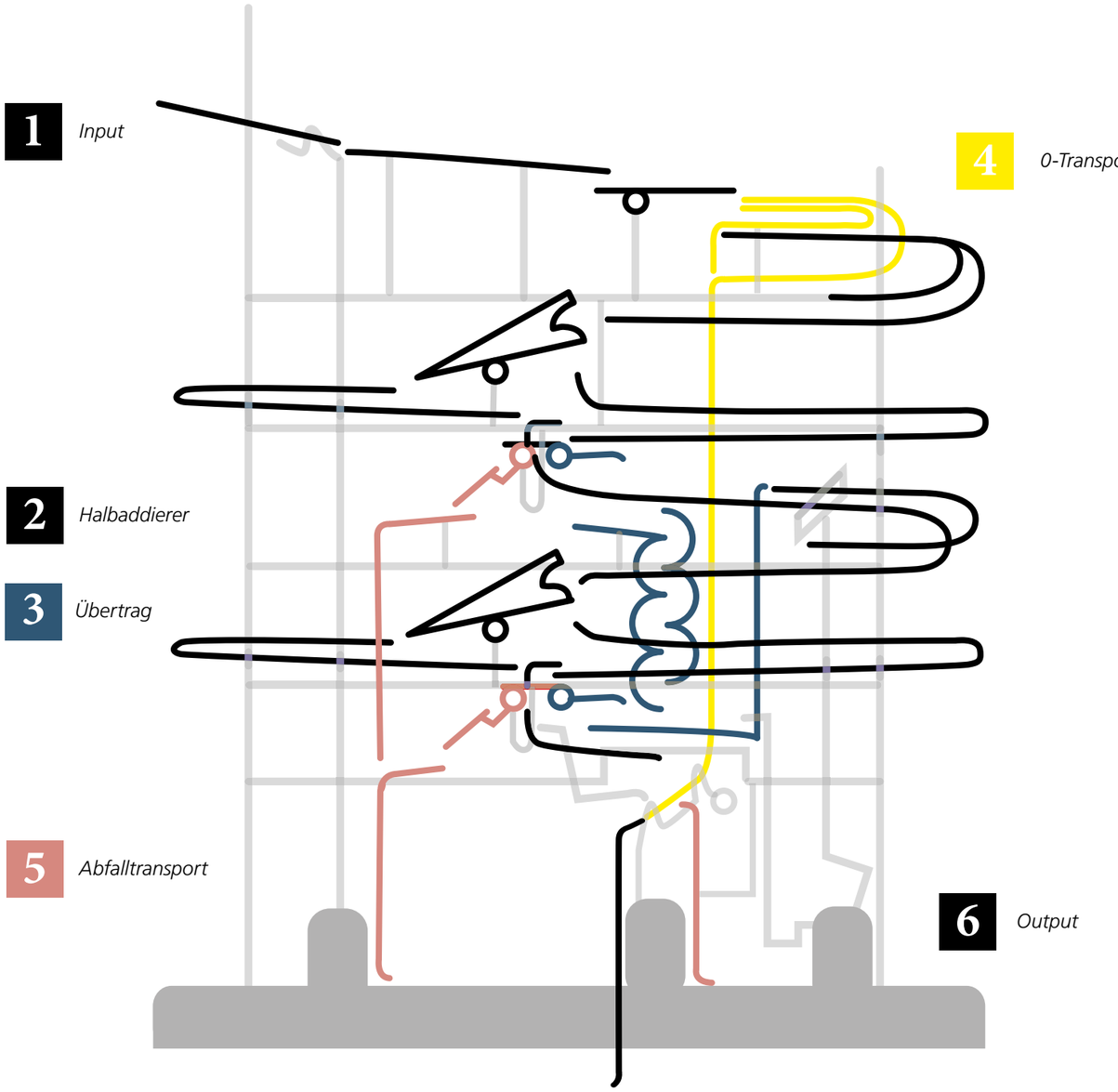
Die Entscheidung, die beiden Zustände eines Bits im Input und Output mit Hilfe von zwei unterschiedlichen physischen Objekten zu realisieren, beeinflusste die Konstruktion der Mechanik maßgeblich. Das Resultat sind einige Unterschiede und Ergänzungen zum bereits gezeigten grundsätzlichen Aufbau:

- Eine direkte 1:1-Umsetzung der elektronischen Logikgatter, aus denen Halbaddierer aufgebaut sind, ist mechanisch nicht möglich, da die physischen Informationsträger im Gegensatz zu elektrischen Signalen nicht gesplittet werden können. Die Logik eines Halbaddierers ist hier deshalb als Gesmates umgesetzt.

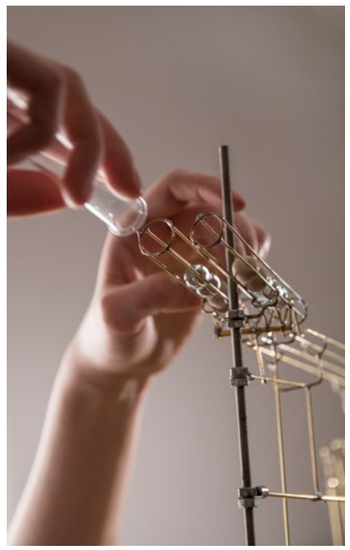
- Die eigentliche Rechenoperation wird nur mit den 1-Murmeln durchgeführt, weshalb die 0-Murmeln zu Beginn aussortiert und direkt über eine Röhre zum Output transportiert werden.

- Da in einigen Fällen eine 0 als Output benötigt wird, aus dem Input jedoch keine 0 eingeht, werden einige 0-Murmeln in der Transportröhre „gespeichert“ gehalten.

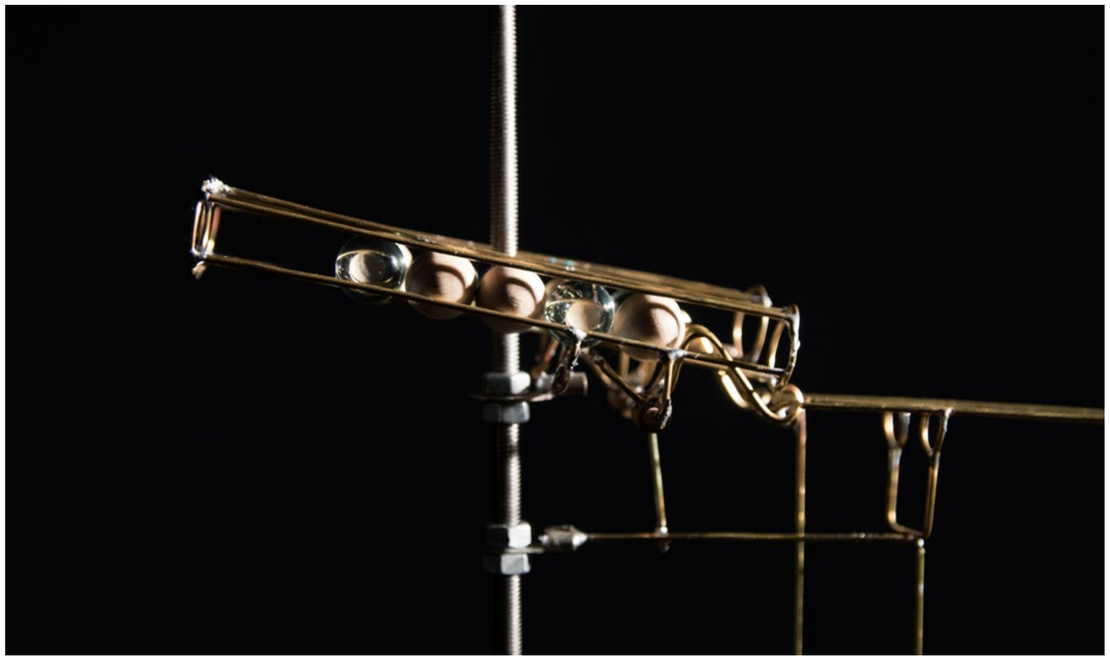
- Bei zwei Eingangs- und einem Ausgangssignal muss einer von zwei eingehenden Informationsträgern notwendigerweise „entsorgt“ werden, weshalb an allen relevanten Stellen Abfall-Transport-Bahnen existieren.



Übersicht aller Bahnverläufe. Kreise markieren bewegliche Scharniere.



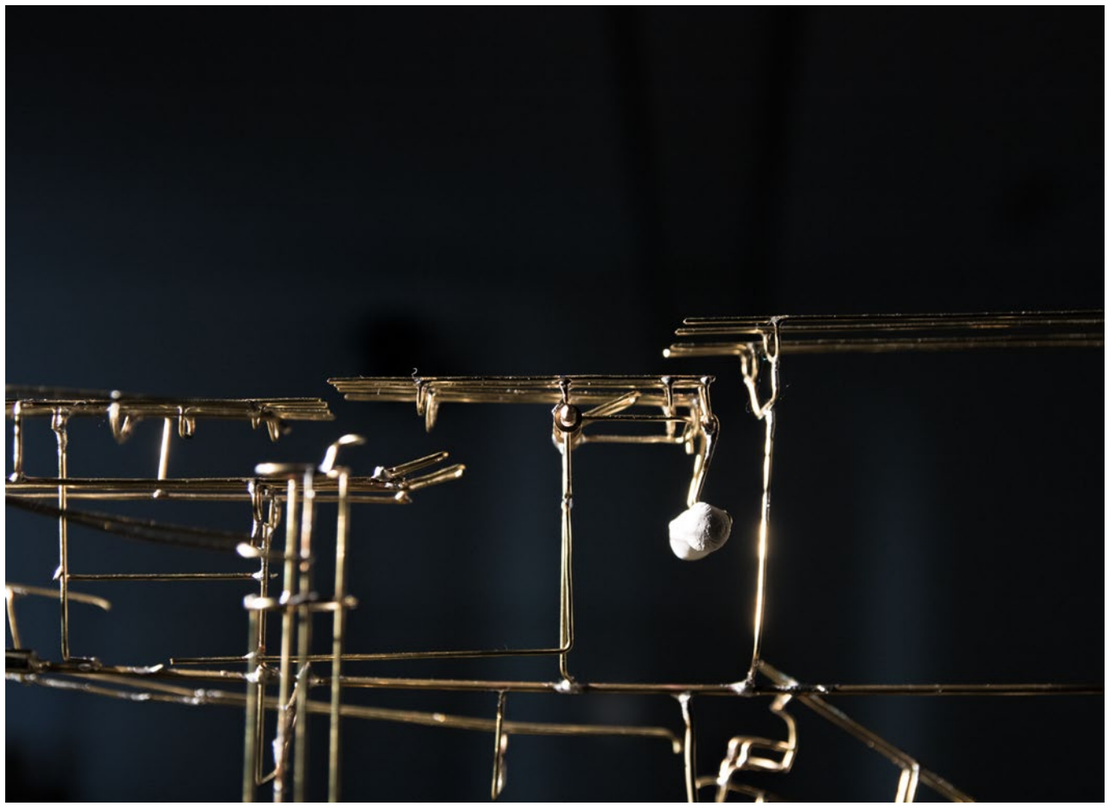
Mit den entsprechenden Dezimalzahlen beschriftete Reagenzgläser dienen als Behältnis für die codierten Binärzahlen



Input-Register befüllt mit dem Wert 10010 = 18

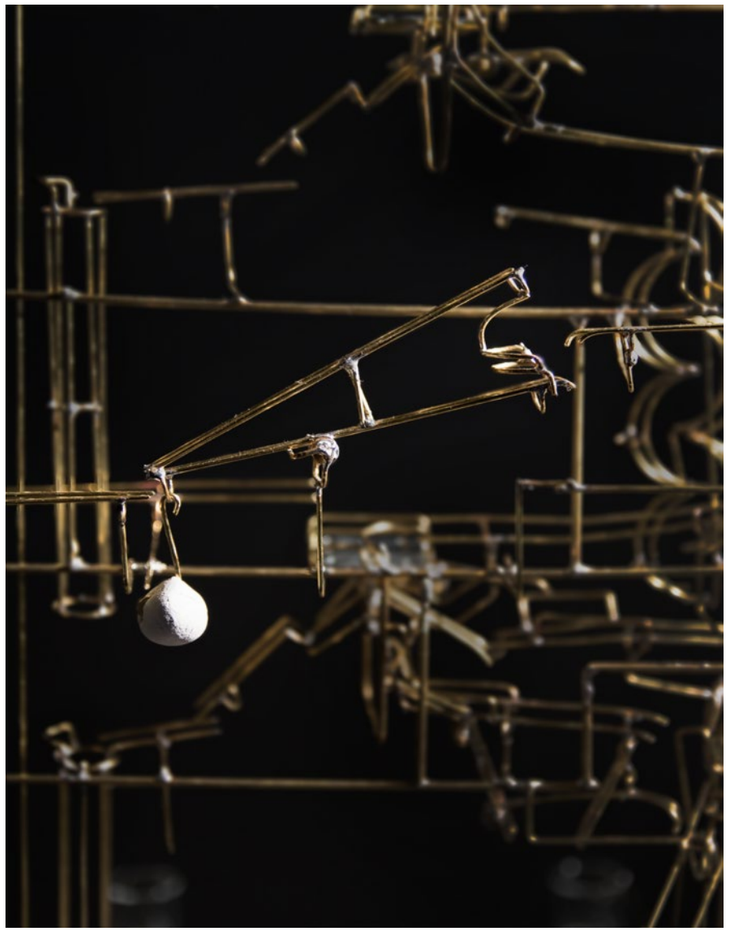
Der Input

Die Werte „1“ und „0“ werden durch schwere Glasmurmeln und leichte Holzkugeln repräsentiert. Zum Rechnen werden die beiden Input-Zahlen mit der letzten Stelle voraus in die Register gegeben. Eine Hebelmechanik lässt mit jedem Takt die jeweils vordersten Kugeln gleichzeitig frei und startet so den Rechenprozess. Zur einfacheren Eingabe der Zahlen in einer Aufstellungssituation werden die Zahlenfolgen im Voraus in beschriftete Reagenzgläser gefüllt, die dann einfach in das Register geschüttet werden können.



Der 0-Transport

Der eigentliche Rechenprozess erfolgt nur über die Anzahl der „1“-Murmeln.
Daher werden die 0-Murmeln direkt zu Beginn über eine auf das leichte Gewicht der Holzkugeln nicht reagierende Wippe aussortiert und über einen Schacht direkt in Richtung Output transportiert. Die schwereren Glasmurmeln lösen die Wippe aus und werden auf eine zum ersten Halbaddierer führende Bahn zusammengeführt.

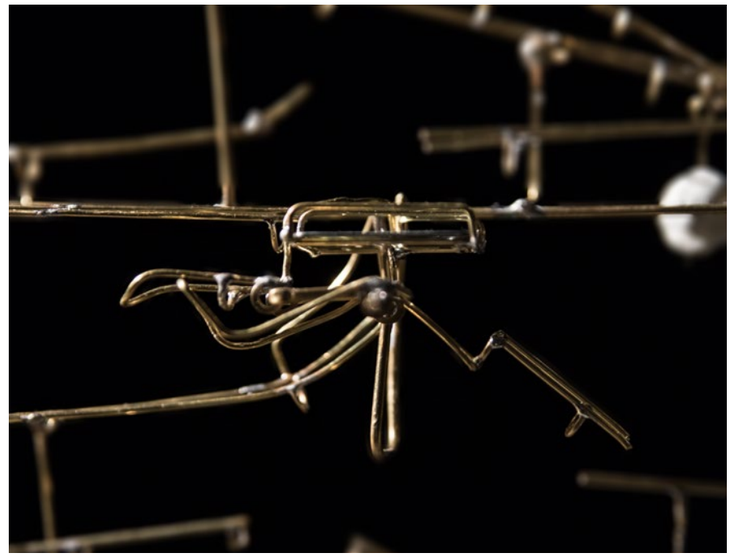


Die Wippe sortiert die eintreffende(n) Murmel(n) auf gegenüberliegende Bahnen, die aus verschiedenen Richtungen zu der Hebelmechanik führen.

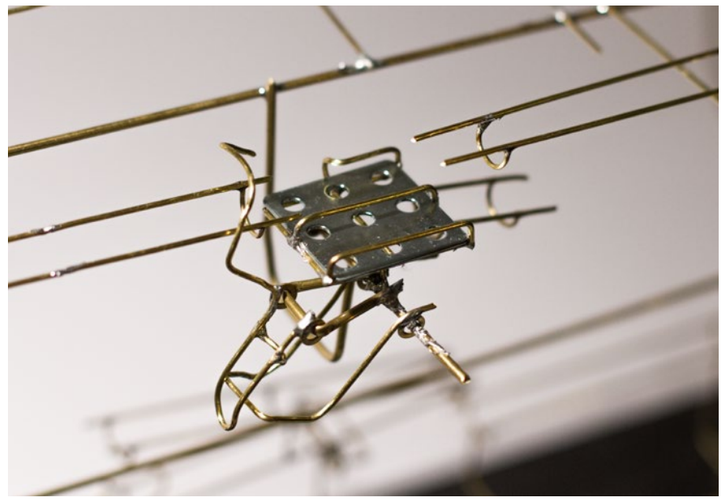
Die Halbaddierer

sind der Ort der eigentlichen Berechnung und damit das Herzstück des Addierwerks.

Die Berechnung erfolgt in zwei Schritten. Eine eintreffende Murmel fällt in den Korb der Wippe, drückt diese nach unten und wird auf die darunter liegende Bahn geleitet. Folgt ihr eine zweite Murmel, wird dieseüber den oberen Pfad der Wippe auf die nach links führende Bahn gelenkt. Ein Gegengewicht setzt die Wippe anschließend zurück.
Im zweiten Schritt löst die eventuell vorhandene zweite Murmel beim Herunterrollen von der silbernen Plattform (Bild rechts unten) einen Hebel aus, der den im 90° Winkel gebogenen Bügel auf der gegenüberliegenden Bahn anhebt und rollt anschließend weiter in den Übertragsspeicher.
Die entgegenkommende erste Murmel kann nun ungehindert über die Plattform in den Abfallschacht rollen und setzt dabei die Mechanik zurück. Somit ist ein Übertrag entstanden.
Gibt es keine zweite Murmel, bleibt der Bügel folglich unten und die erste Murmel wird nach links abgelegt und zum zweiten Halbaddierer weitergeleitet. Es ist kein Übertrag entstanden.



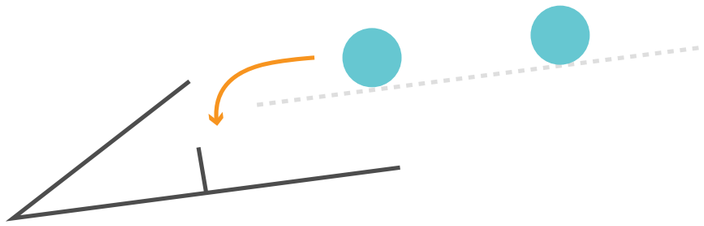
Die Hebelmechanik in der Profilansicht. Der linke Hebel führt zum Übertragsspeicher, der rechte in den Abfallschacht. Die mittlere Bahn dahinter führt in den zweiten Halbaddierer.



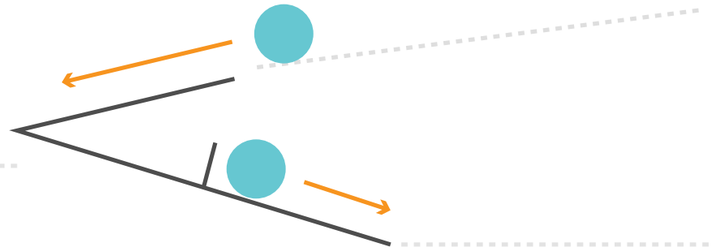
Eine von rechts kommende zweite Kugel hat den Hebel nach unten gedrückt, sodass die von links kommende Kugel unter dem Bügel durchrollen kann.

Funktionsschema der Halbaddierer-Mechaniken

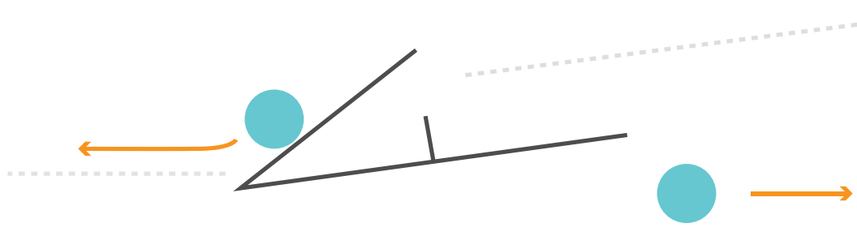
1



2

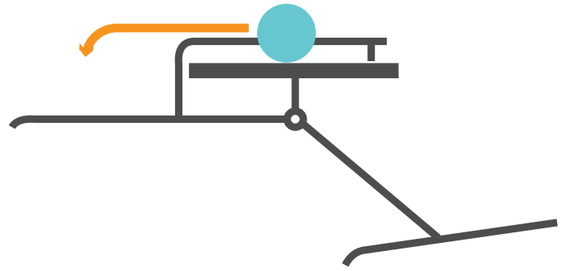


3

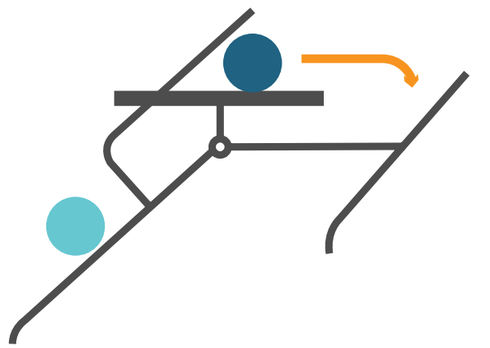


Verteilung von zwei Einsen auf entgegengesetzte Bahnen

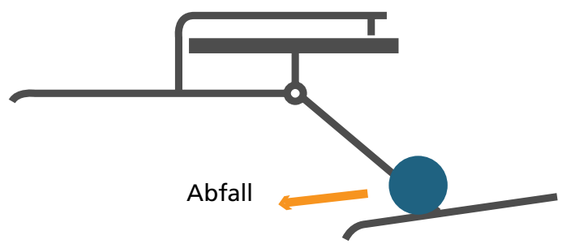
1



2

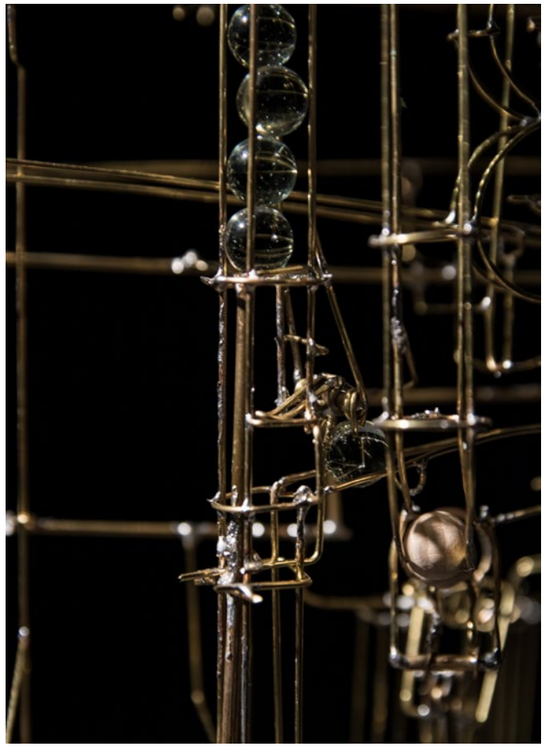


3

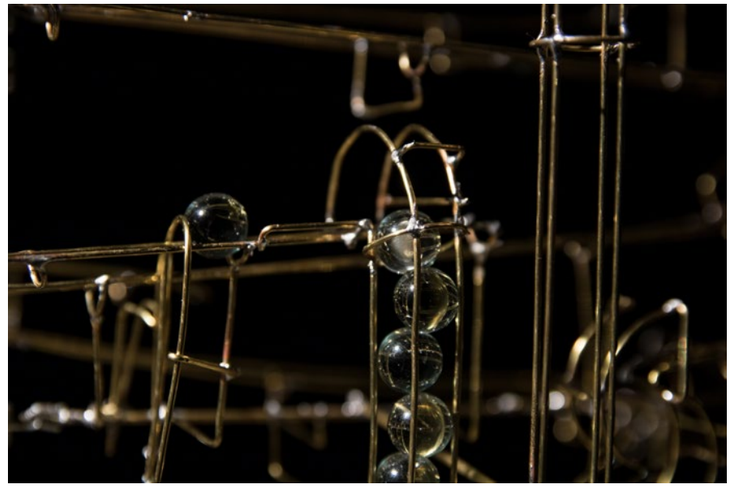


Umlenkung von zwei Einsen in Übertrag und Abfall

Die Übertragsmechanik im inaktiven Zustand. Die verschachtelten Bögen (rechts im Bild) bremsen den Fall von Murmeln aus dem ersten Halbaddierer.



Die Übertragsmechanik im aktiven Zustand. Die eintreffende Murmel wird zurückgehalten, bis der Hebel in der zweiten Taktphase wieder gesenkt wird.



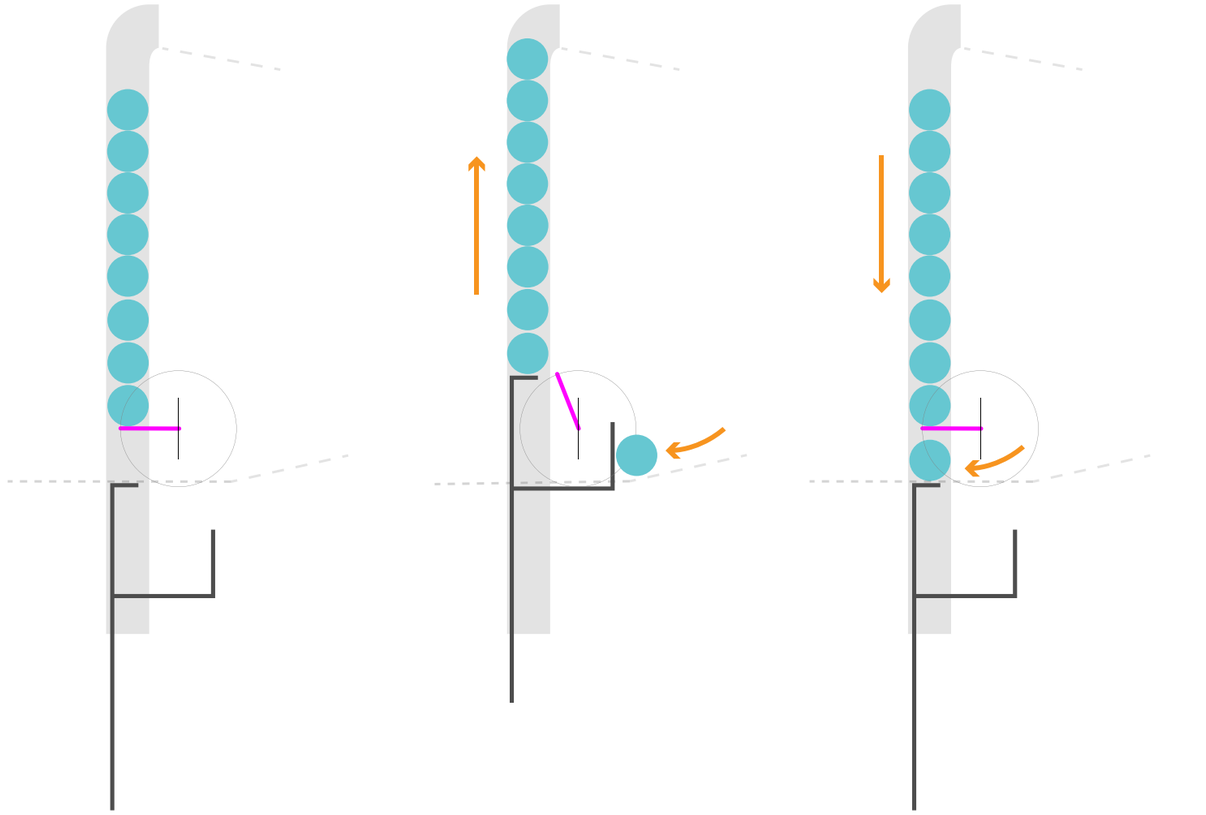
Ein Buffer fängt die Murmeln nach dem Austreten aus dem Speicher ab

Die Übertragsmechanik

Die Übertragsmechanik ist durch einen mit 1-Murmeln gefüllten Schacht realisiert. Zu Beginn jedes Taktes wird der gesamte Murmelstapel durch einen Hebel angehoben. Eingehende Überträge rollen unten in den Schacht. Ist im vorausgehenden Takt ein Übertrag entstanden, wird der Stapel durch die zusätzliche Kugel so weit nach oben gedrückt, dass die oberste Murmel aus dem Speicher geschoben wird: Der Übertrag wird geladen.

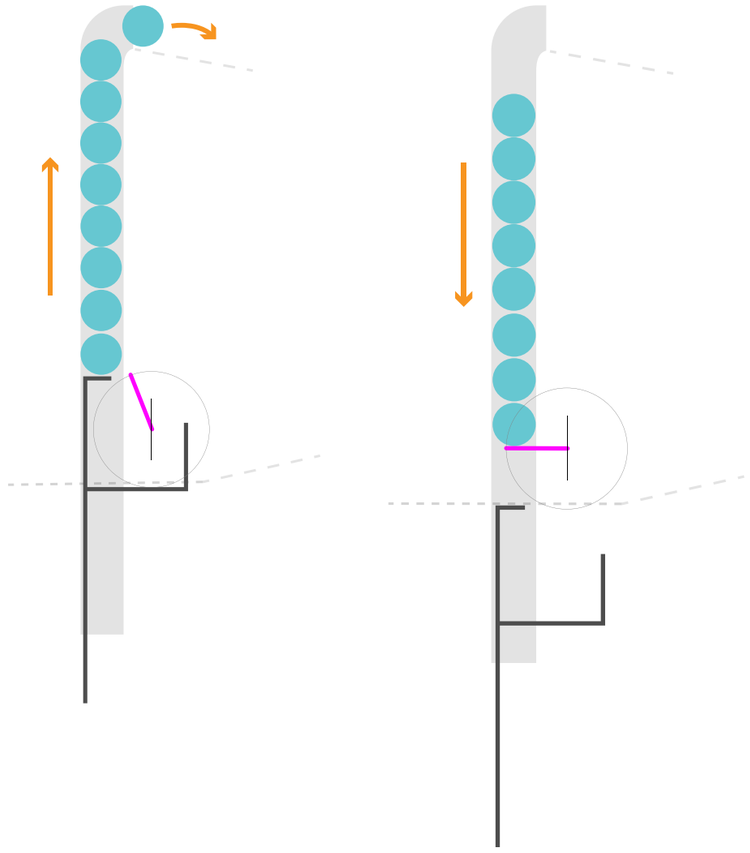
Damit der Übertrag und die Summe aus dem ersten Halbaddierer synchron in den zweiten Halbaddierer laufen, werden beide zunächst von einem Buffer, der die Bahn in der ersten Taktphase schließt, aufgehalten. In der zweiten Phase des Taktes werden sie gleichzeitig freigegeben.

Speicherung und Ausgabe eines Übertrags



Takt 1: Speichern eines Übertrags

Die Höhe des Stapels reicht nicht aus, um eine Murmel aus dem Schacht herauszuschieben: der Übertrag ist „0“.
Während sich der Übertrag im aktiven Zustand befindet, wird der eintreffende Übertrag zurückgehalten.

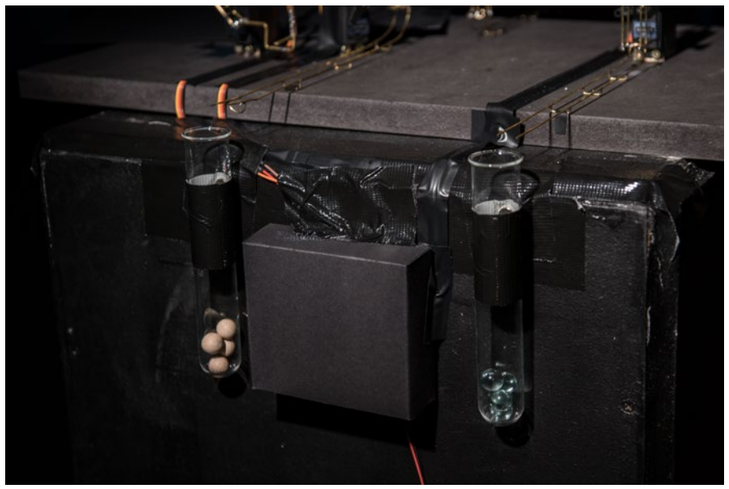


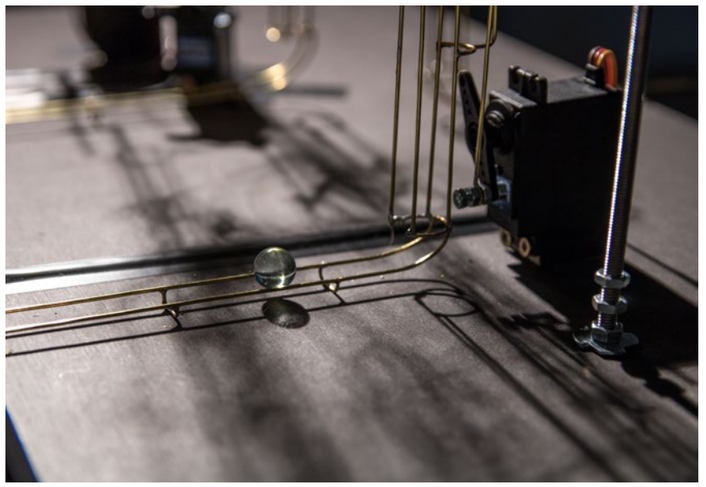
Takt 2: Ausgabe eines Übertrags

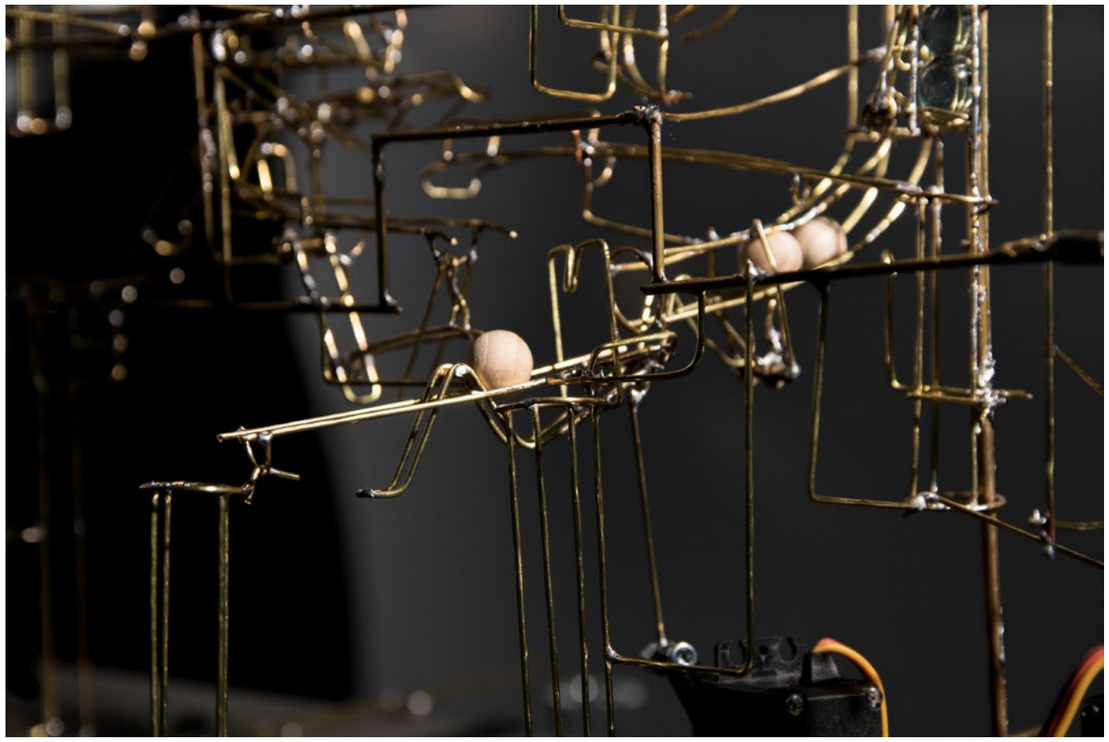
Im aktiven Zustand ist der Stapel durch die zusätzliche Murmel hoch genug, um eine Murmel oben aus dem Schacht herauszuschieben.

Der Abfalltransport

Aussortierte Murmeln werden in über einen Schacht in auf der Rückseite befindliche Behälter transportiert. Es gibt separate Auffangröhren für nicht benötigte Nullen am Output und für aussortierte Einsen aus den Halbaddierern.







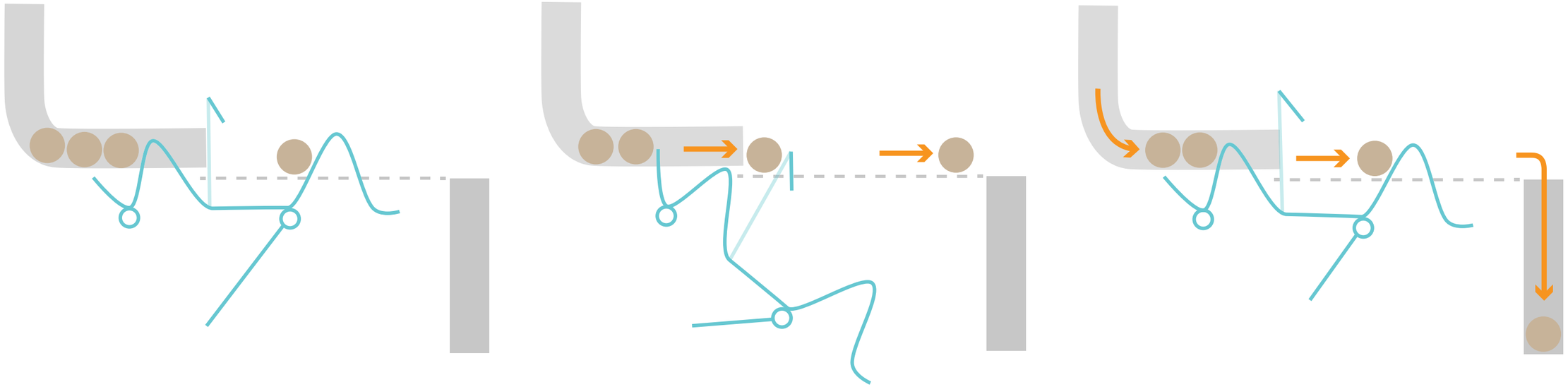
Eine Holzmurmel am Ausgang der Murmelbahn. Wird sie nicht von einer Glasmurmel verdrängt, wird sie im nächsten Takt in den Ergebnisspeicher transportiert.

Der Output

Am Ausgang der Murmelbahn befindet sich ein Hebel, der sich zu Beginn jedes Taktes öffnet und die am Ausgang liegende Murmel in den Ergebnisspeicher befördert.

Aus dem 0-Speicher wird gleichzeitig eine Null herausgelassen, die in der zweiten Takthälfte den Platz der vorherigen Murmel einnimmt. Ist nun aus der Berechnung keine 1 hervorgegangen, wird die 0 im nächsten Takt in den Output-Speicher geleitet. Ist jedoch eine 1 hervorgegangen, betätigt diese auf dem Weg nach unten einen Hebel, der als eine Art Schreibkopf fungiert. Dieser stößt die am Ausgang liegende 0 aus der Bahn. Die 1 nimmt anschließend deren Platz ein. Im nächsten Takt wird damit eine 1 in den Output befördert.

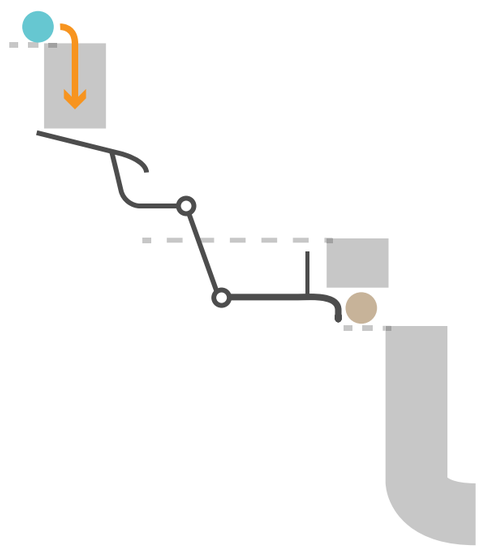
Funktionsschema des Output-Hebels



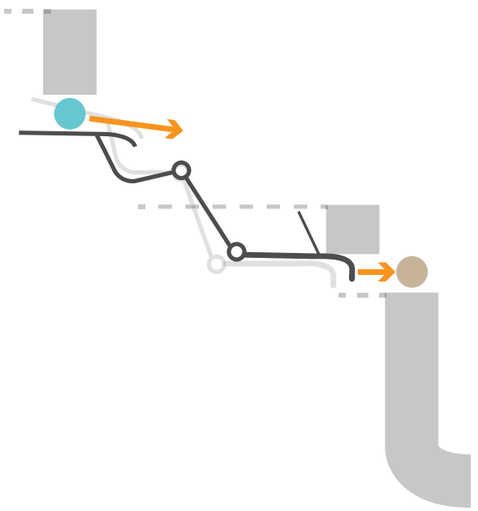
links: Hebelposition „geschlossen“
mitte: Position „geöffnet“ in Taktphase 1
rechts: Position „geschlossen“ in Taktphase 2

Funktionsschema des 0 / 1 -“Schreibkopfs“

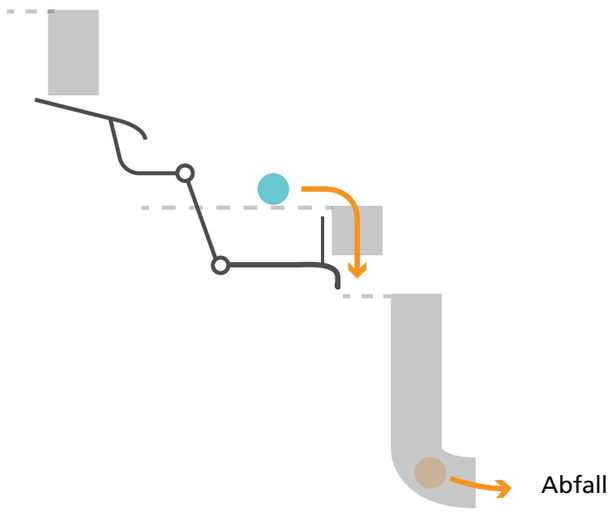
1



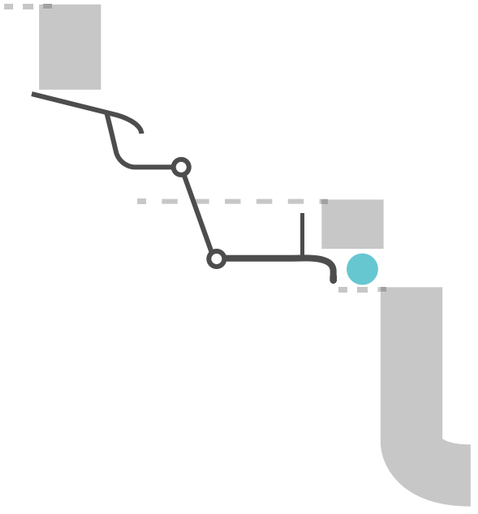
2

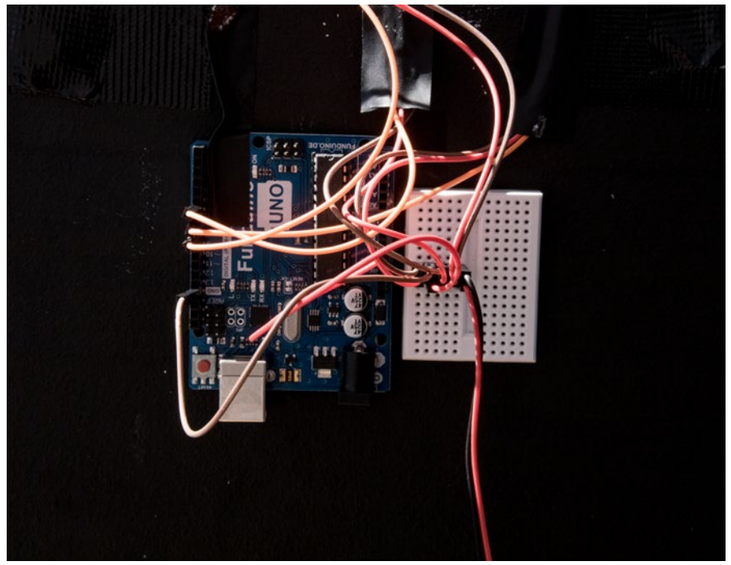


3

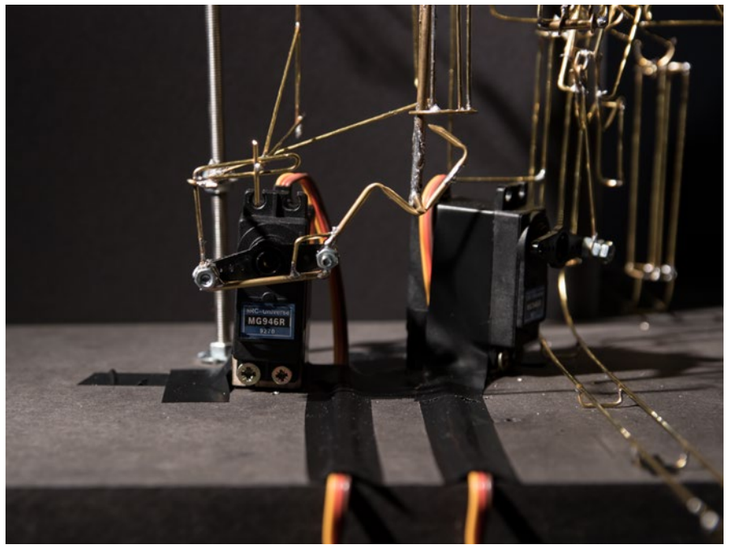


4





Die Steuerungselektronik des Taktsignals

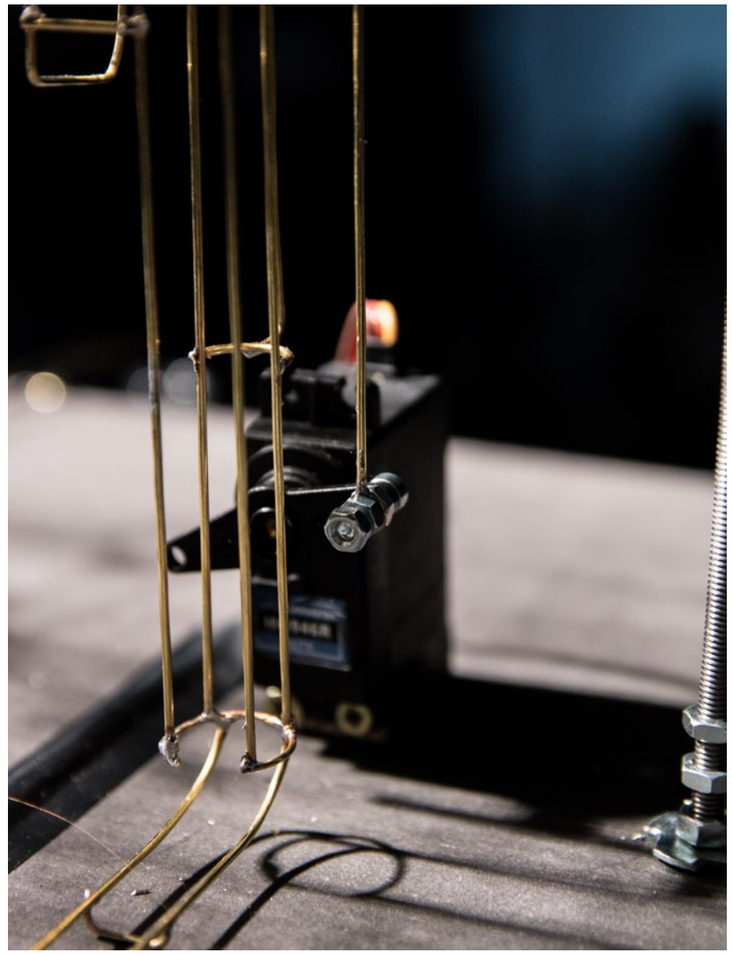


An die Übertragsmechanik (links) und den Output (rechts) angeschlossene Motoren

Der Signaltakt

Das Taktsignal erfolgt über drei an der Bodenplatte festgeschaubte Motoren, die durch einen Arduino angesteuert werden. Die Taktlänge des Prototypen betrug 10 Sekunden. Version 2 hat eine Taktlänge von 13 Sekunden.

Die Taktung der Motoren ist zweiphasig: in der ersten Phase werden alle angeschlossenen Bauteile in den geöffneten bzw. aktiven Zustand versetzt. Nach der Hälfte der Periodendauer werden alle Elemente zurückgesetzt. Dies dient dazu, die Murmeln bzw das Ergebnis aus der ersten Addition zu puffern, um es in der zweiten Taktphase mit dem Output aus dem Übertrag zu verrechnen.



Motor zum Öffnen und Schließen der Input-Register

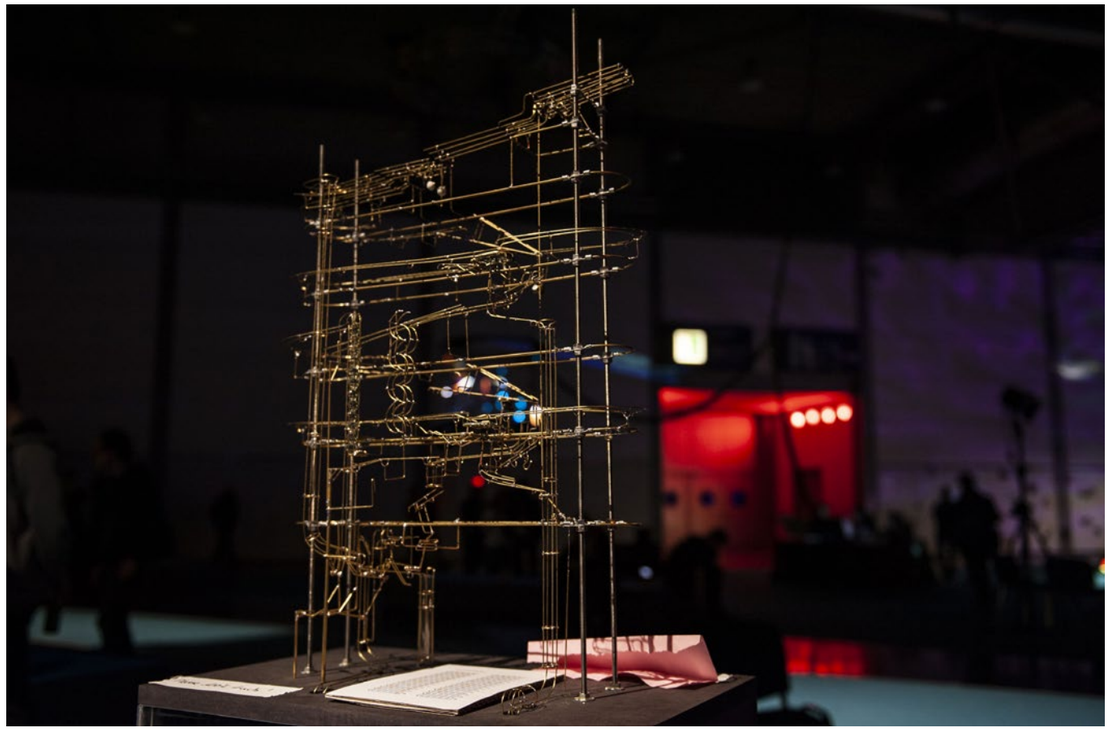
Programmcode zur Steuerung der Servomotoren

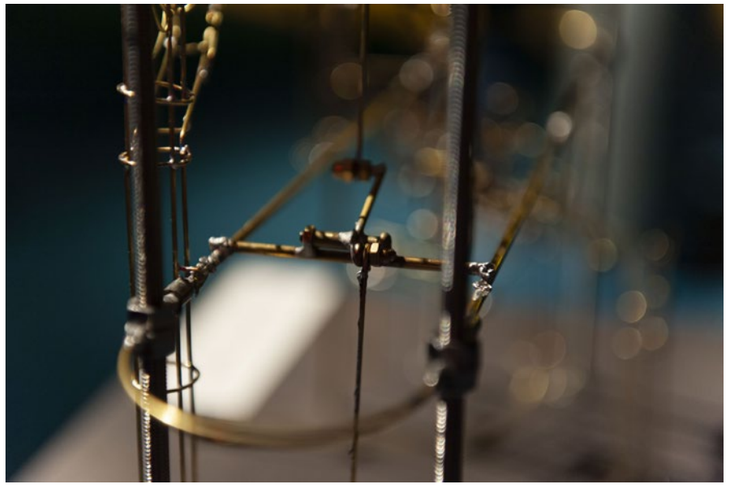
```
#include <Servo.h>
#include <Metro.h>

Servo servo1, servo2, servo3;

// servo positions
int servo1_off = 75;
int servo1_on = 50;
int servo2_off = 170;
int servo2_on = 100;
int servo3_off = 160;
int servo3_on = 30;
int half_period = 6500;
bool isOn = false;

Metro metro = Metro(half_period);

void setup()
{
    //attach servos to pins and set position to zero
    servo1.attach(6); // Input
    servo2.attach(7); // Ouput
    servo3.attach(8); // Carry + Buffer
    servo1.write(servo1_off);
    servo2.write(servo2_off);
    servo3.write(servo3_off);
}

void loop()
{
    if (metro.check() == 1)
    {
         if (isOn) {
          servo1.write(servo1_off);
          servo2.write(servo2_off);
          servo3.write(servo3_off);
          isOn = false;
         } else {
          servo1.write(servo1_on);
          servo2.write(servo2_on);
          servo3.write(servo3_on);
          isOn = true;
         }
    }
}
```

Neubau des Rechenwerks

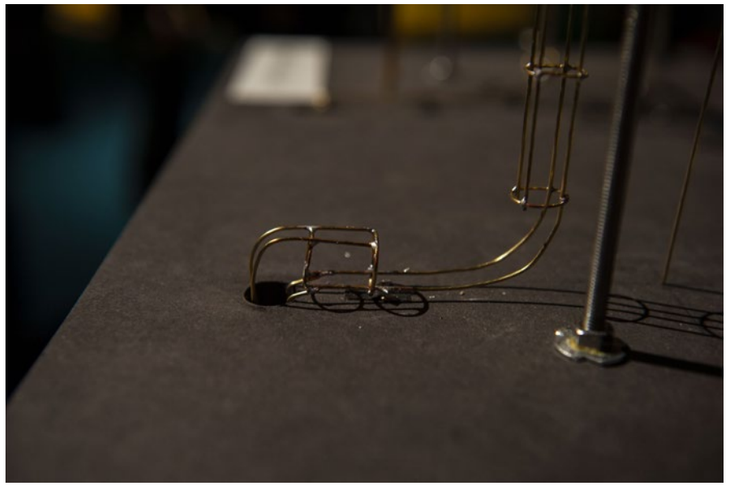
Bei längerem Betrieb des Prototypen zeigte sich, dass dessen Gerüst durch die Fixierung an nur zwei Punkten recht instabil war und dadurch das ganze Modell in jedem Takt durch die Bewegung der Motoren erschüttert wurde. Zudem erwiesen sich einige Bauteile insbesondere in Bezug auf das richtige Timing der Murmeln als äußert fehleranfällig, was Anlass zu einem Neubau gab.

Die zweite Version des Rechenwerks ist in ihrem grundsätzlichem Aufbau und Funktionsweise mit dem Prototypen identisch, enthält aber zahlreiche Verbesserungen der Mechnik, die für eine geringere Fehlerquote und höhere Stabiltät sorgen.

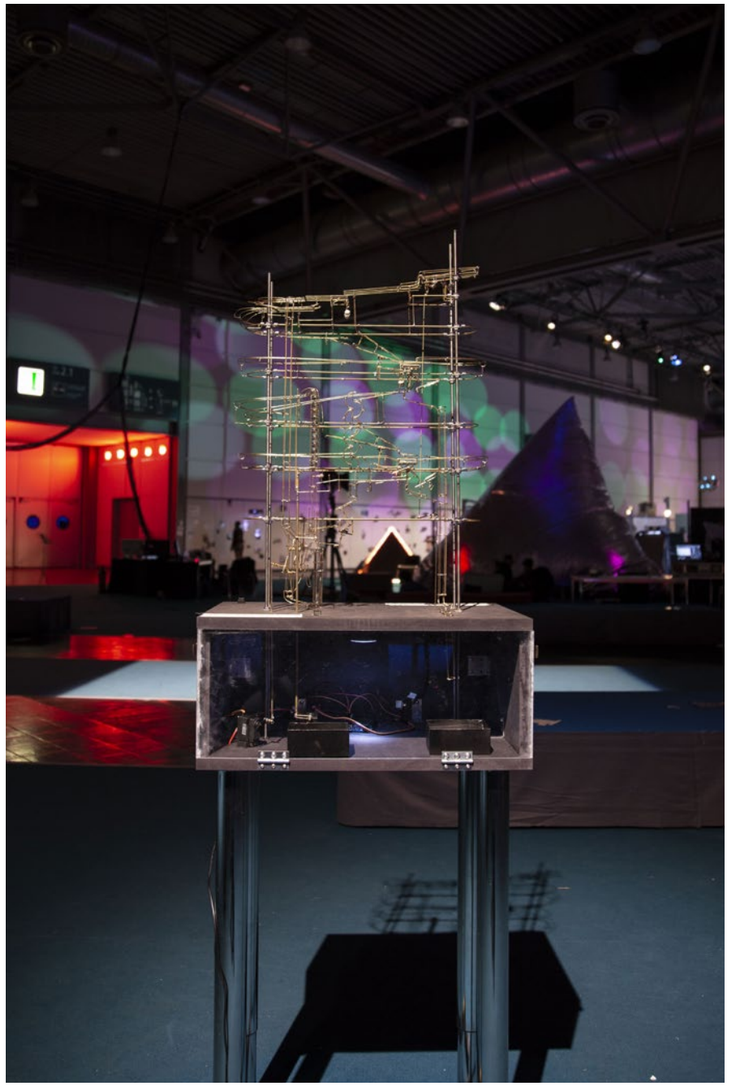




Hebel zur Übertragung der Motorenbewegung unterhalb der Inputregister



Enstehende „Abfälle“ werden nun ins Innere des Kastens geleitet

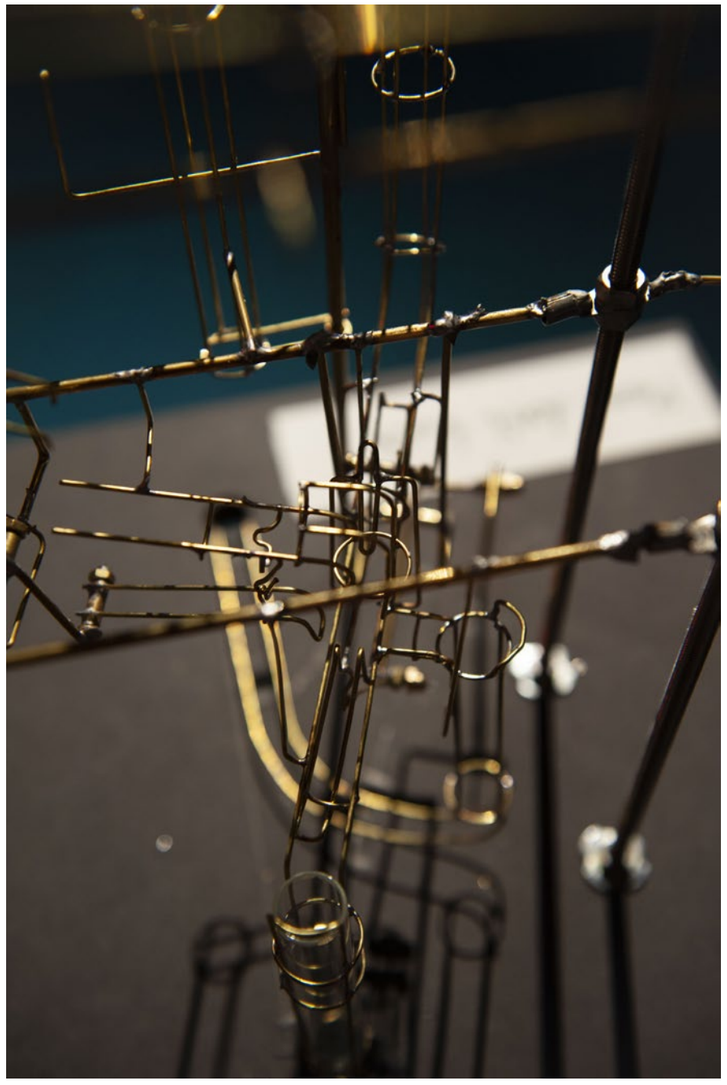


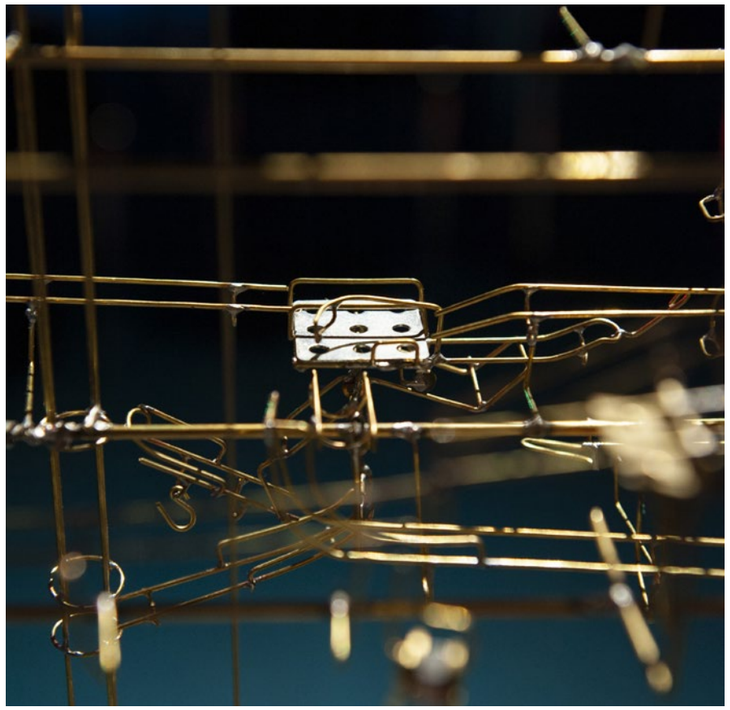
Rückansicht des Modells

Der größte Verbesserungsbedarf bestand bei allen an die Motoren angesschlossenen Komponenten: Die Umlenkung der Motorenbewegung über zusätzliche Hebel sorgt an Input und Output für sanftere Schließbewegungen, während der Hub-Arm des Übertrags größere Stabilität durch seine Führung durch die Bodenplatte erhält. Zudem wurde die Buffermechanik deutlich verinfacht.

Die auffälligste Neuerung ist die Verlagerung der Servomotoren ins Innere des Kastens, auf den dieBodenplatte aufgeschraubt ist. Dieser besitzt eine Klappe aus Plexiglas, wodurch Motoren und Elektronik in den Hintergrund treten, aber trotzdem sichtbar bleiben. Auch die Abfalltransporte führen ins Innere des Kastens. Dadurch ergeben sich im Vergleich zum Prototypen größere Gesamtmaße.

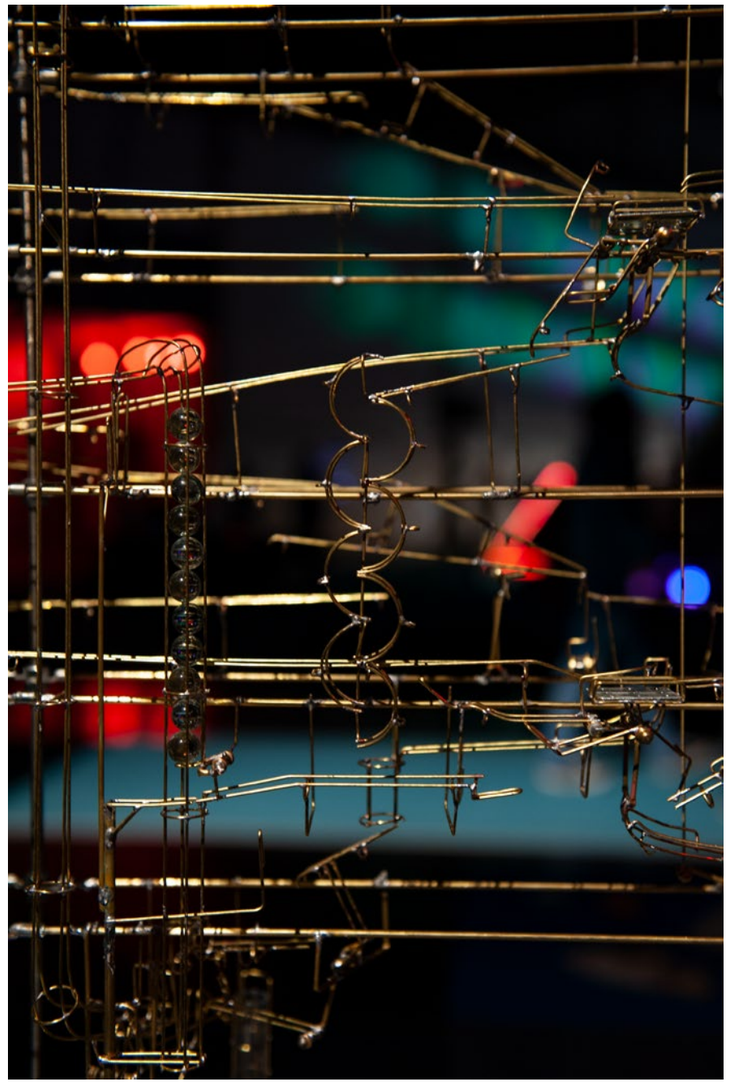
Außerdem verfügt der Neubau über ein stabileres Gerüst aus vier Gewindestangen, die auf fünf Ebenen über Messingrohre miteinander verbunden sind.





Verbessertes Halbaddierer-Bauelement

Geringfügige Veränderungen des Layouts sind die Platzierung des Ergebnisspeichers auf der Bodenplatte und die positionierung des Übertragsspeichers rechts von der 0-Röhre.
Zudem erhielten die beiden Bauelemente der Halbaddierer mechanische Verbesserungen für ein zuverlässigeres Timing der Murmeln.



Technische Details

Daten zu Version 2 des Rechenwerks

Maße

Kasten: 55 x 35 x 25 cm
Gerüst: 32 x 6.5 x 71 cm
Gesamt: 55 x 35 x 96 cm

Material:

Kasten: 19 mm schwarze MDF
Klappe: 0.25 cm Plexiglas

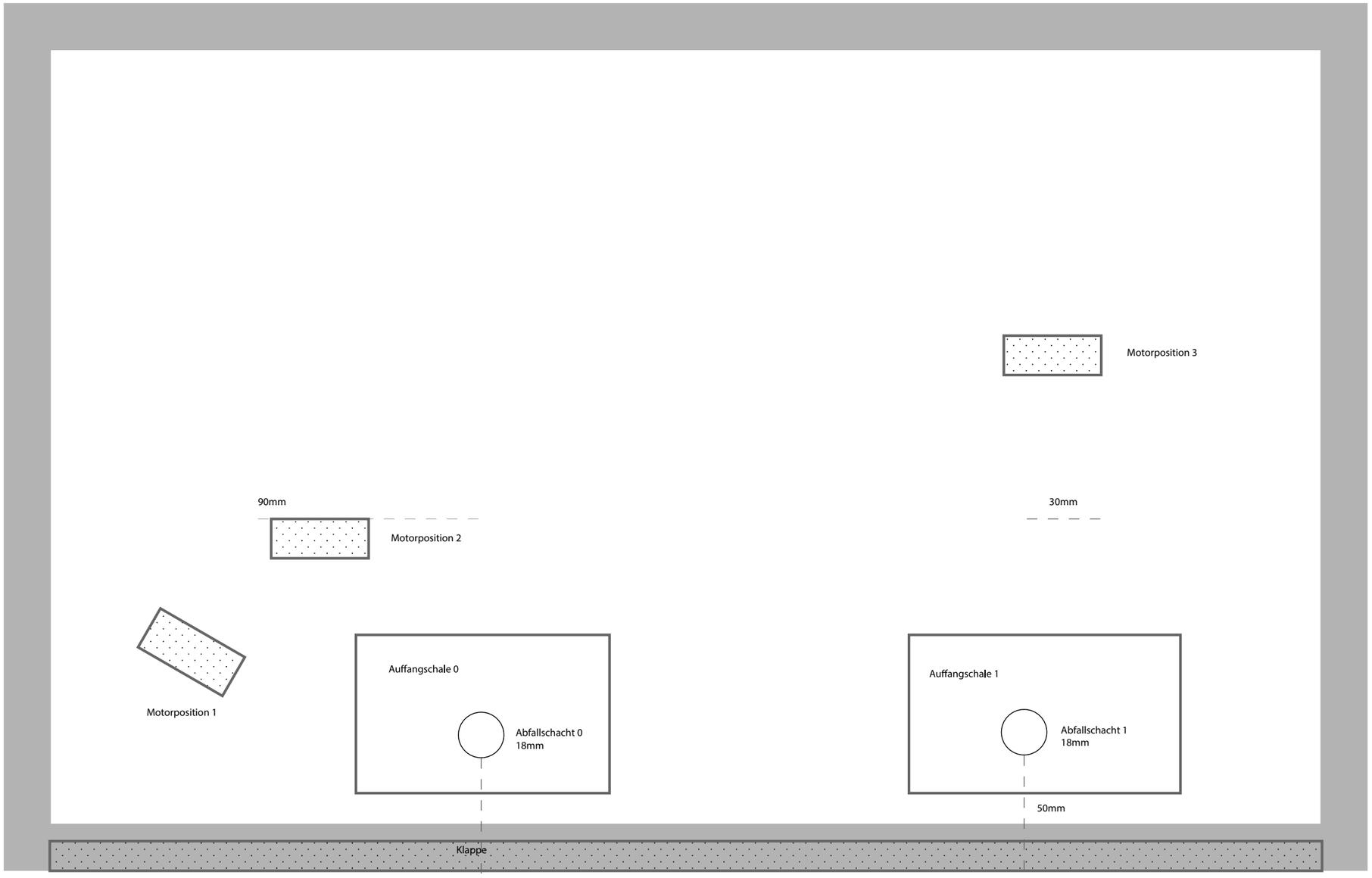
Gerüst Längsstreben: 6 mm Stahlgewinde
Gerüst Querstreben: 4 mm Messingrohr
Gerüst Verbindungen: 6,4 mm Kabelschuhe
Außerdem: 6mm Muttern, 6 mm Einschlagmuttern

Bauteile: 1,0 mm und 1,2 mm Messingdraht (verdrillt); 2mm Messingrohr
Gelenke: 4 mm Messingrohr (außen); 2,5 mm Messingschrauben (innen)

Murmeln – „1“ : 12 mm Glasmurmeln, 2,3g
Murmeln – „0“: 12 mm Holzmurmeln , ca. 0.3 g

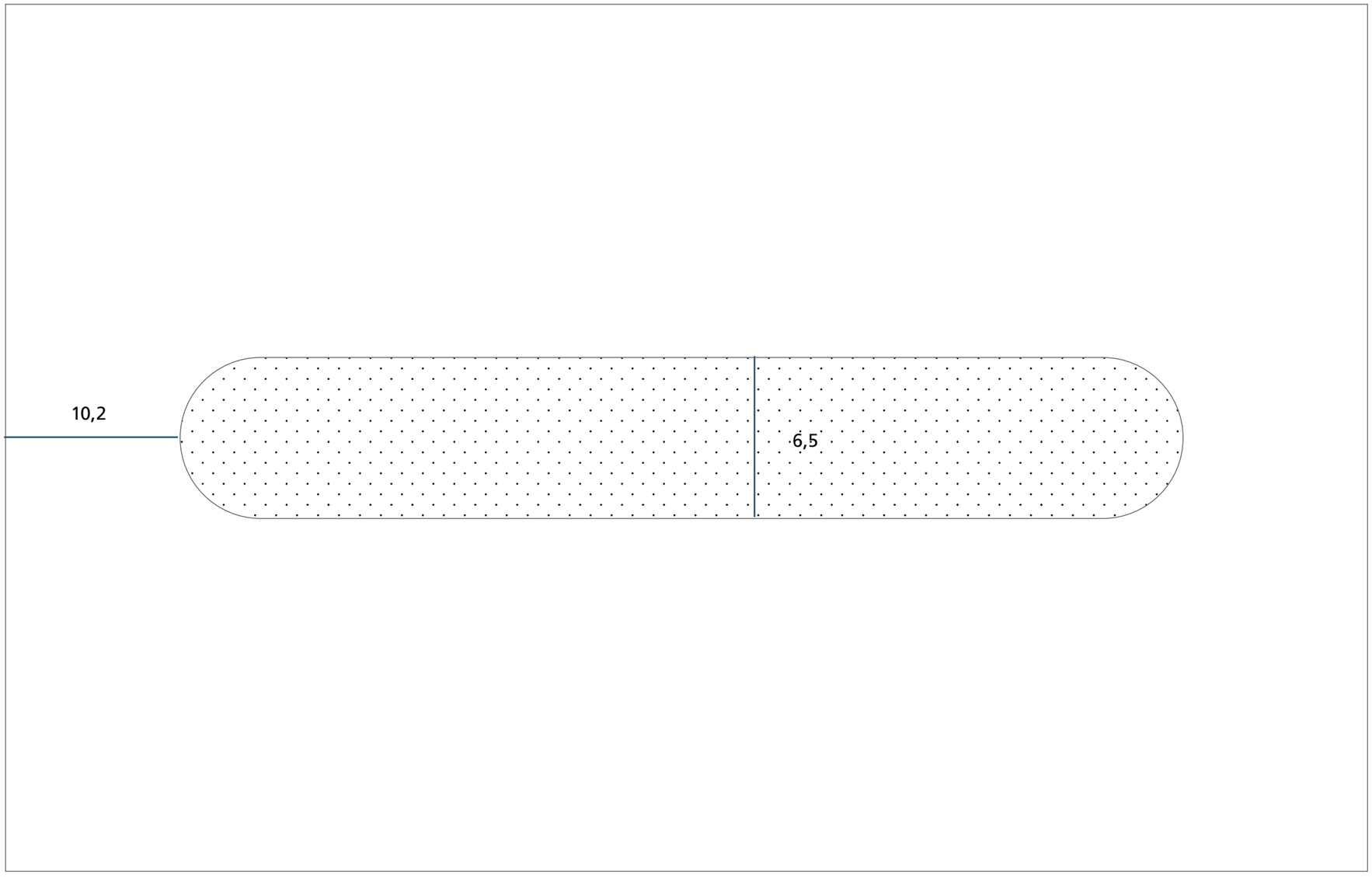
Elektronik:

Controller: Arduino Uno R3
Motoren: MG946R digitale Servomotoren, Stellkraft 12kg
Stromversorgung: 6V Netzteil
Taktfrequenz: 1/13 Hz



Grundfläche 55,0 x 35,0

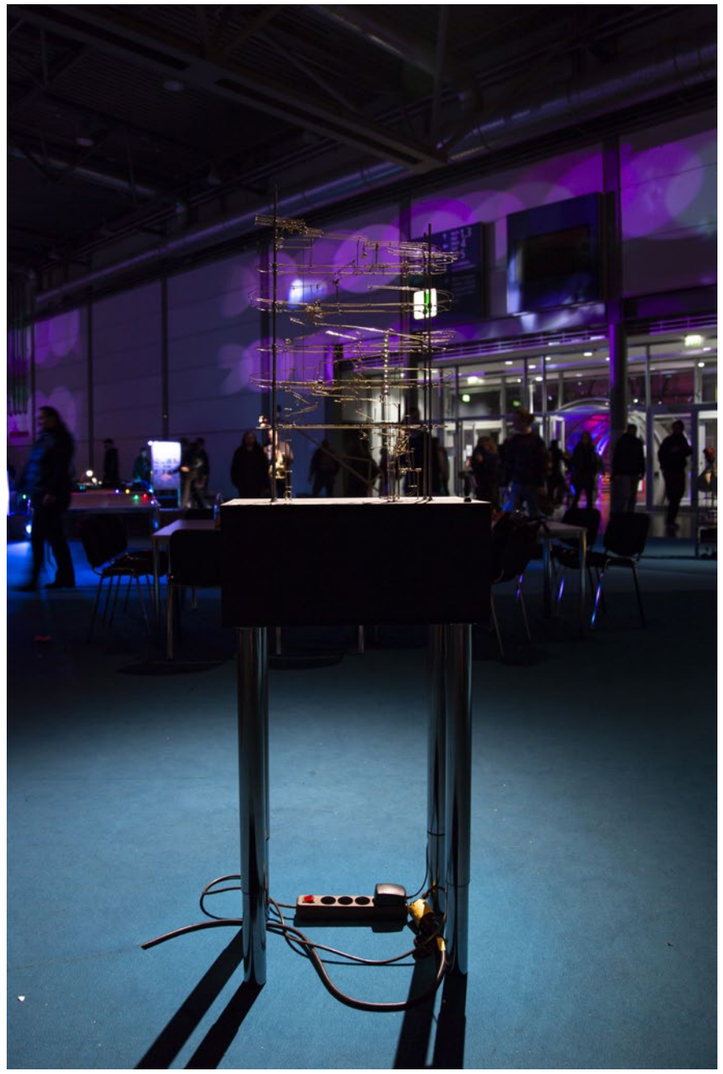
Layout der unteren Bodenplatte

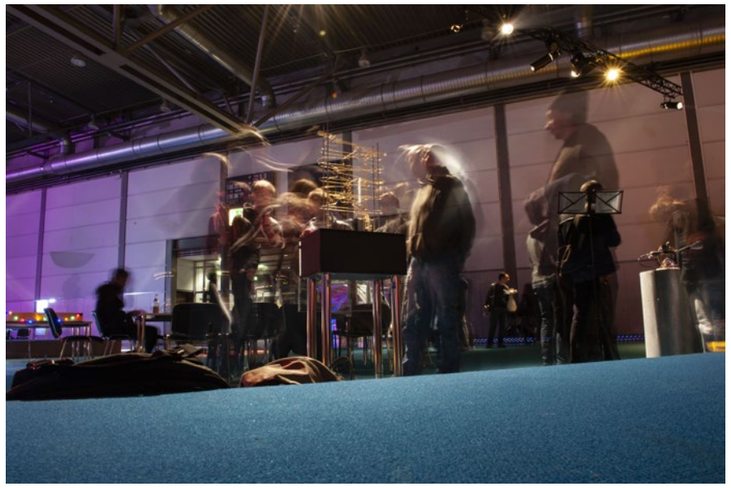


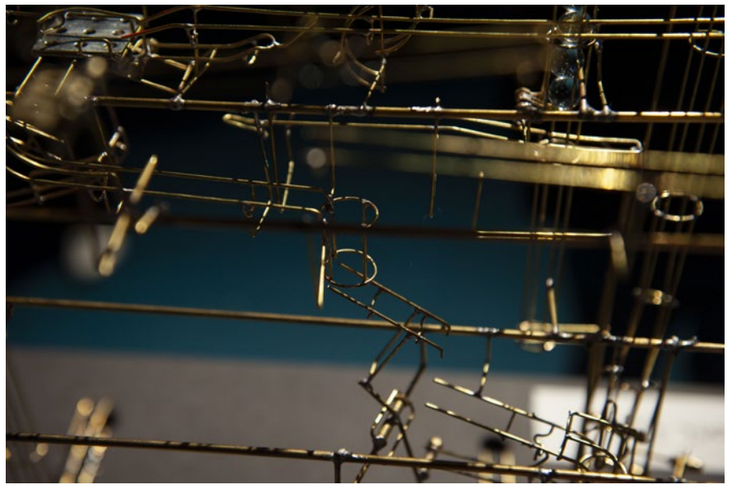
Grundfläche 55,0 x 35,0

Platzierung des Gerüsts auf der oberen Bodenplatte

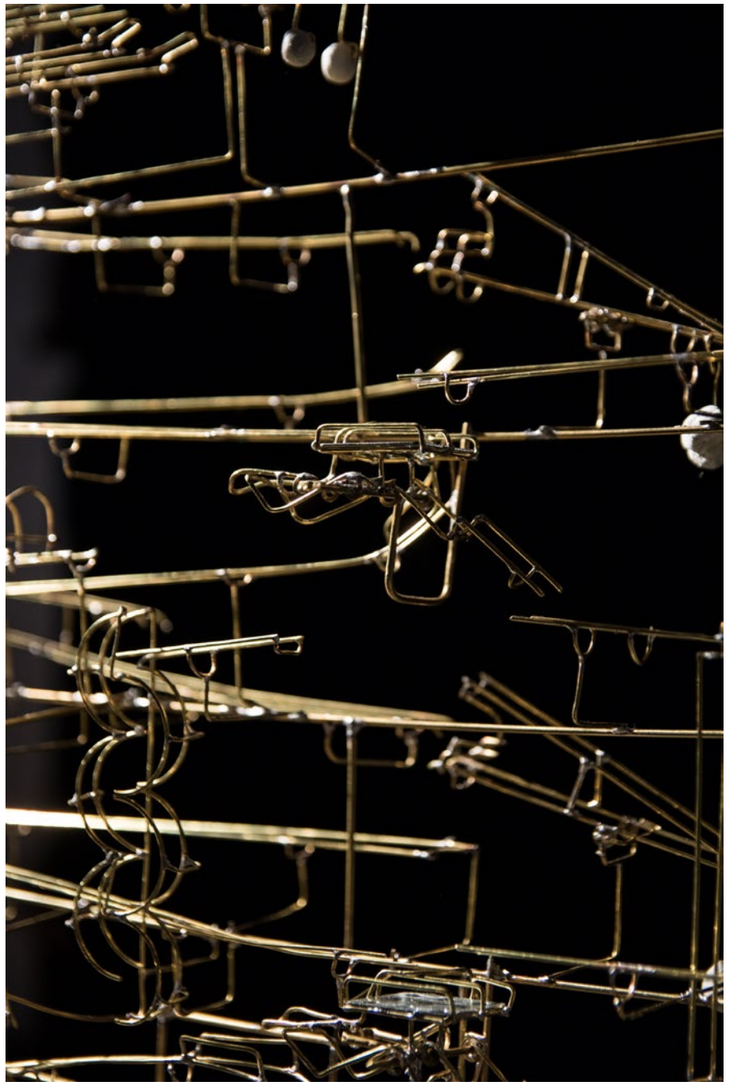
Weitere Bilder

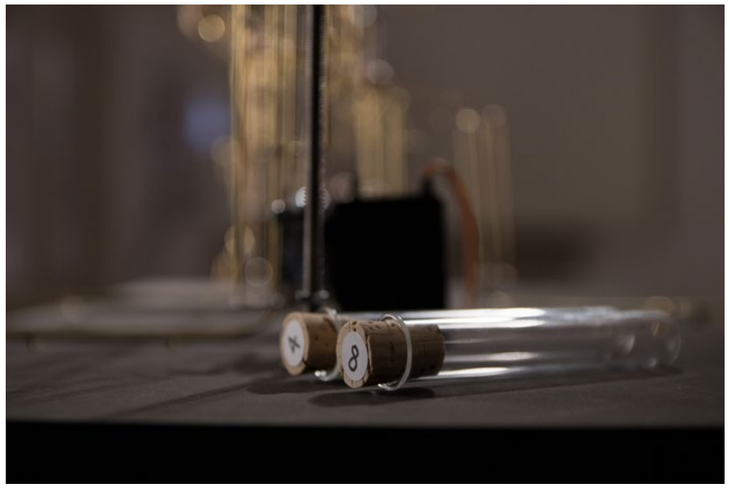


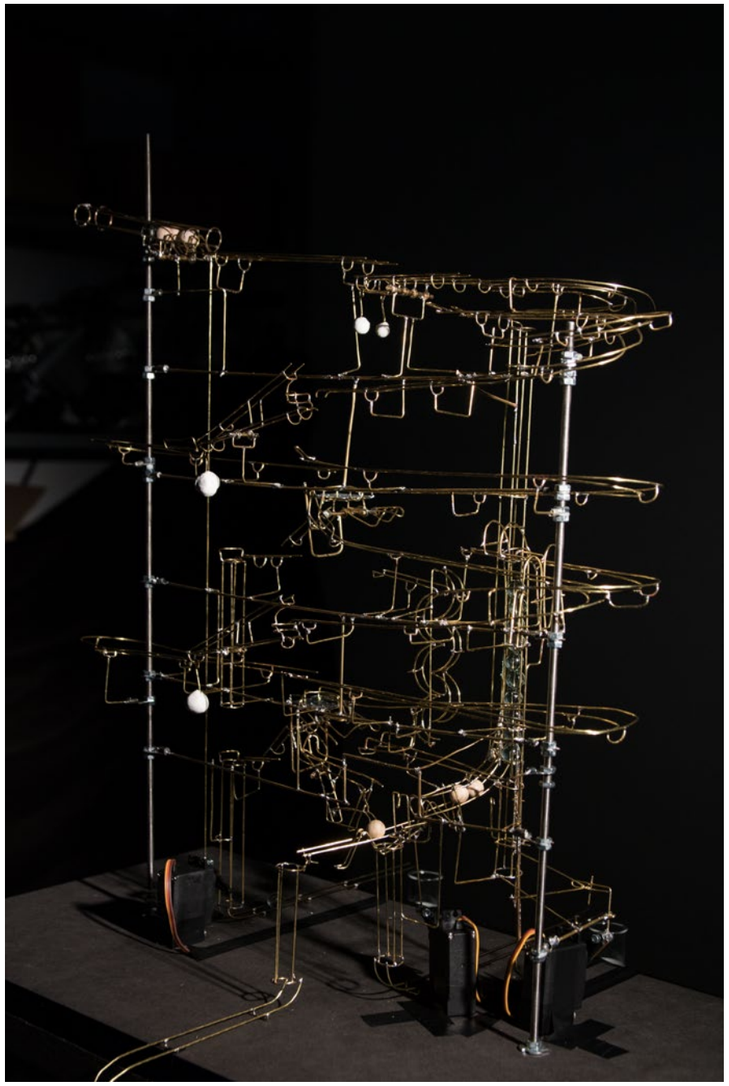




Version 2 im Art and Play - Space des 35C3







Weitere Bilder des Prototypen

Kontakt

Franziska Schneider
E-Mail: franziska.schneider@me.com
Mobil: +49 152 216 44869