Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.

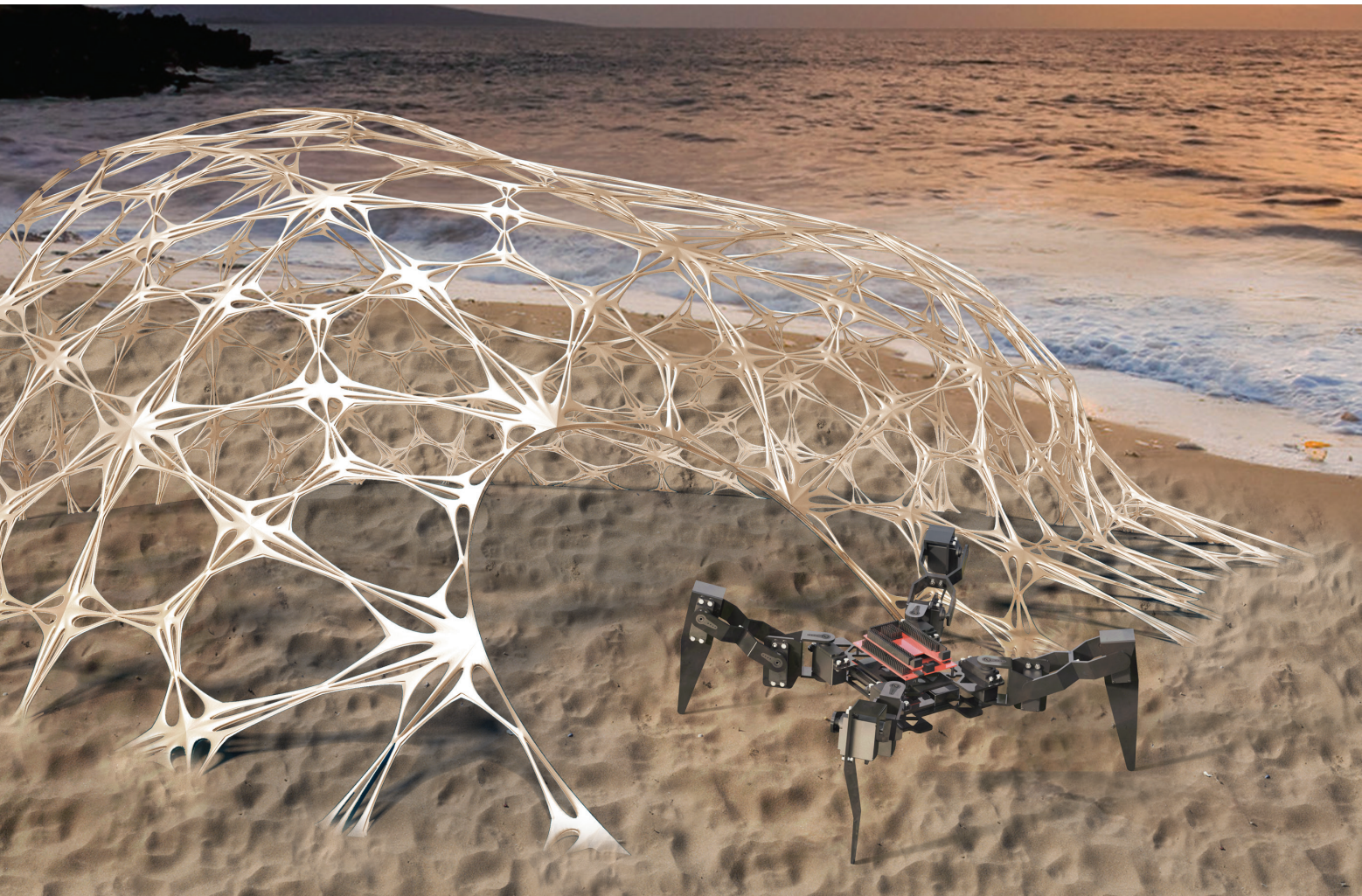
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

roboterarchitektur

der bauarbeiter der zukunft



TU UB

Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.

http://www.ub.tuwien.ac.at

TU WIEN UB Universitätsbibliothek

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

MASTER-/DIPLOMARBEIT

Roboterarchitektur.

Der Bauarbeiter der Zukunft

Robotic Architecture.

Construction worker of the future

ausgeführt zum Zwecke der Erlangung
des akademischen Grades eines
Diplom-Ingenieurs / Diplom-Ingenieurin
unter der Leitung von

Manfred Berthold
Prof Arch DI Dr

unter Mitbetreung von

Christoph Müller
Univ Lektor DI Dr

E253 - Institut für Architektur und Entwerfen

eingereicht an der Technischen Universität Wien
Fakultät für Architektur und Raumplanung

Christoph Kolbeck
Matr. Nr. 01328028

A 1050 Wien
Storkgasse 6/42

+43 680 2201970
christoph.kolbeck@me.com

Wien, am _____
Datum

_____
Unterschrift

abstract

Abstract DE

Ein Swarm Robotic System setzt sich aus vielen kleinen Einheiten (Agenten) welche über Entscheidungs- und Handlungsmöglichkeiten verfügen. Das System-Verhalten ergibt sich dabei aus dem Verhalten der einzelnen Agenten und wird nicht von einem übergeordneten System kontrolliert.
Dies soll im Rahmen meiner Diplomarbeit anhand von wandelbarer, selbstorganisierender Architektur untersucht werden. Hierbei entwickeln spinnenähnliche, steigfähige Roboter (Agenten) eine Struktur aus Fasern, welche sie selbst erzeugen. Die daraus entstandenen Strukturen können bei neuen Anforderungen umgestaltet bzw. auch wieder rückgebaut werden.
Die Agenten stellen den „Bauarbeiter der Zukunft" dar.

Ein besonderes Augenmerk liegt auf der Nachhaltigkeit. Analog zu dem Begriff Urban Mining erfolgt hier die Materialgewinnung in Form eines „Waste Mining". Die Agenten werden an verschmutzten Stränden ausgesetzt und sammeln und sortieren dort den angespülten Müll. Aus dem verwertbaren Kunststoff wird dann druckbares Filament generiert. Nach einer Art Bedürfnispyramide entwickeln sie dann Strukturen die zunächst zum Sortieren des Mülls (Körbe) bis hin zu von Menschen und Tieren nutzbaren Architekturen reichen.
Ziel der Diplomarbeit ist eine Simulation des Entstehungsprozesses und die Visualisierung der generierten architektonischen Strukturen mittels 3D-Druck sowie der Bau eines Prototyps.

Abstract EN

A Swarm Robotic System consists of many small units (agents) which have decision and action options. The system behavior results from the behavior of the individual agents and is not controlled by a higher-level system.

This will be examined in the context of my diploma thesis on the basis of changeable, self-organizing architecture. Here spider-like robots (agents) develop a structure of fibers, which they produce themselves. The resulting structures can be redesigned or even dismantled to meet new requirements.
The agents represent the „construction worker of the future".

A special focus is on sustainability. Analogous to the term urban mining here the material extraction takes place in the form of a „waste mining". The agents are exposed on polluted beaches and collect and sort there the washed-up garbage.
From the recyclable plastic printable filament is then generated. After a kind of need pyramid, they then develop structures that initially range from the sorting of garbage (baskets) to buildings that can be used by humans and animals.
The aim of the diploma thesis is a simulation of the development process and the visualization of the generated architectural structures by means of 3D printing as well as the construction of a prototype.

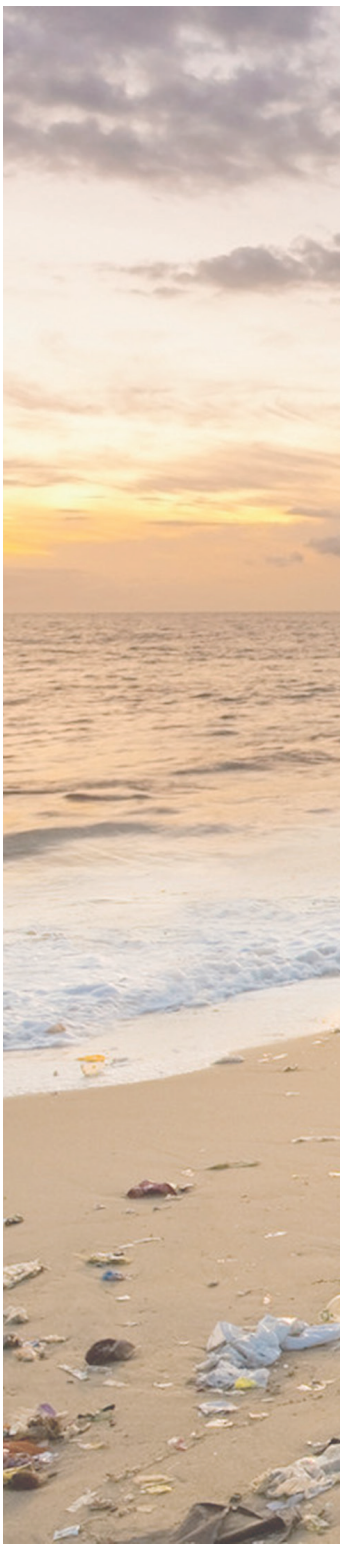
index

1 intro 8
2 situationsanalyse 10
2.1 plastik problematik 12
2.2 robotik in der architektur 16
2.3 verortung 18
3 ziele der arbeit 22
4 methodik 24
4.1 einstieg 26
4.2 exkurs: victimless crime 30
4.3 quadropoden 32
4.4 inverse kinematik 34
4.5 roboter funktionsschema 36
4.6 software controller 38
4.7 software prozessor 40
4.8 ablaufdiagramm 42
4.9 formgeneration 44
4.10 systematik 46
4.11 roboterentwicklung 48
4.12 prototyping 56
4.13 werkzeuge 58
4.14 sensoren 60
4.15 kunstoffarten 62
4.16 druckexperimente 64
5 analyse/synthese 100
6 ergebnis 102
6.1 prototyp 104
6.2 pläne 106
6.3 fotos 108
7 bewertung 112
8 zusammenfassung 114
9 verzeichnisse 120
9.1 code 124
10 curriculum vitae 148
11 danksagung 149

1 intro

*„Früher war die Erde einmal ohne Plastik.(...)
Nach der Steinzeit, der Bronze- und der Eisenzeit haben wir jetzt die Plastikzeit.“* (Werner Boote)

Im Jahr 2014 wurden 3110000000 Tonnen Plastik hergestellt. Nur ein verschwindend geringer Teil davon wird recycelt. Plastikmüll stellt deshalb eine enorme Herausforderung unserer Zeit dar. Viele Organisationen und Projekte beschäftigen sich mit dem Sammeln und Recyceln dieses Mülls. Während der Versuch die anfallenden Massen an Plastikmüll zu beherrschen eher einer Sysiphosaufgabe gleicht, könnte der Einsatz autonomer Roboter unter diesen Voraussetzungen einen wesentlichen Vorteil bringen. Über einen längeren Zeitraum eingesetzt, können die Roboter somit einerseits Raum wieder nutzbar machen und andererseits, Rohstoffe wieder einem Recyclingkreislauf zugeführt werden.
In der vorliegenden Arbeit wird ein Szenario kreiert, indem autonome Maschinen über mehrere Jahre an einem verschmutztem Strand ausgesetzt werden. Anschließend beginnen sie damit den Strand aufzuräumen, die Materialien zu sortieren und wiederzuverwerten um schließlich neue, nutzbare Strukturen für Menschen und Tiere zu erstellen. Die autonomen Maschinen stellen so etwas wie den „Bauarbeiter der Zukunft“ dar, die Strukturen in Form von Architektur selbst generieren können.



Werner Boote in „Plastic Planet“, 2014

Abbildung 1.1 Verschmutzer Strand

2 situationsanalyse

2.1 plastik problematik

Obwohl die es nahezu an jedem Fleck der Erde Probleme mit dem produzierten Plastikmüll gibt, sind einige Effekte besonders hervorzuheben. Ein Großteil des Plastiks gelangt über die Flüsse in die Weltmeere und sammelt sich dort in von Meeresströmungen ausgelösten Strudeln, sogenannten Gyren. So sind in den letzten Jahrzehnten riesige Müllinseln entstanden. Erreicht der Plastikmüll wieder die Küsten, wird er zur zunehmenden Gefahr für die Tierwelt. Auch für den Menschen stellen die riesigen Mengen an Müll eine Gefahr da, da sich das Plastik durch Erosion und UV-Strahlung immer weiter verkleinert und so schlussendlich auch in die Nahrungskette gerät.

Das World Economic Forum hat in seiner im Januar 2016 veröffentlichten Studie „The New Plastic Industry - Rethinking the future of Plastic" Möglichkeiten aufgezeigt wie eine neue Kunststoffindustrie gedacht werden kann. Die Basis dafür stellt eine Umstellung vom bisherigen „linear economy" (Diagramm 2.1-1) hin zu einer „circular economy" (Diagramm 2.1-1) dar. Im Moment werden 95% der Plastikverpackungen nur einmal genutzt, damit entstehen neben einem erheblichen wirtschaftlichen Verlust von 80-120 Milliarden US-Dollar jährlich Umweltschäden in Höhe von geschätzten 40 Milliarden US-Dollar. Die Studie definiert die „circular economy" in drei wesentlichen Punkten:
1. Schaffung einer effektiven „After-Use" Kunststoffwirtschaft. Verbesserung der Wirtschaftlichkeit und Akzeptanz von Recycling, Ausweitung der Einführung von Mehrwegverpackungen und Verwendung kompostierbarer Verpackungen für gezielte Anwendungen.

2. Drastische Reduzierung der sogenannten „leakage" von Kunststoffen (z. B. Kunststoffe, die in den Ozeanen enden) durch Verbesserung der Nachnutzung in den Zielländern, Erhöhung der wirtschaftlichen Attraktivität die Rohstoffe im System zu halten und die Reduzierung der negativen Auswirkungen von Kunststoffverpackungen wenn sie aus dem System fallen.

3. Entkoppelung der Kunststoffe von fossilen Rohstoffen. In erster Linie durch die Entwicklung von Kunststoffen aus erneuerbaren Quellen, wie z.B. biobasierte Kunststoffe oder Kunststoffe welche durch Rückgewinnung aus Treibhausgasen erzeugt werden können.

Darüberhinaus werden Punkte aufgezeigt sie erreicht werden kann:
1. Zusammenarbeit: Einrichtung eines globalen Kunststoffabkommens und Koordinierung von groß angelegten Pilot- und Demonstrationsprojekten unter Beteiligung aller Beteiligten entlang der Wertschöpfungskette, einschließlich politischer Entscheidungsträger und Unternehmen.

2. Innovation: Schaffung großflächiger, zielgerichteter „Moon Shot" Projekte, fortgesetzte Entwicklung und Förderung wissenschaftlicher Forschung in diesem Bereich, wie die Suche nach „Superpolymeren" die Funktionalität mit hervorragender Recyclingfähigkeit kombinieren, sowie umweltfreundlicher Materialien.

3. Kommunikation: Koordinierung und Förderung der Kommunikation über das Wesen des heutigen Systems, die Vision der neuen Kunststoffwirtschaft, Best Practices und Erkenntnisse.

Die vorliegende Arbeit soll ein solches „Moon Shot" Projekt darstellen, indem es das Problem mithilfe von innovativen Roboteragenten adressiert.

The New Plastics Economy - Rethinking the future of plastics, World Economic Forum, 2016

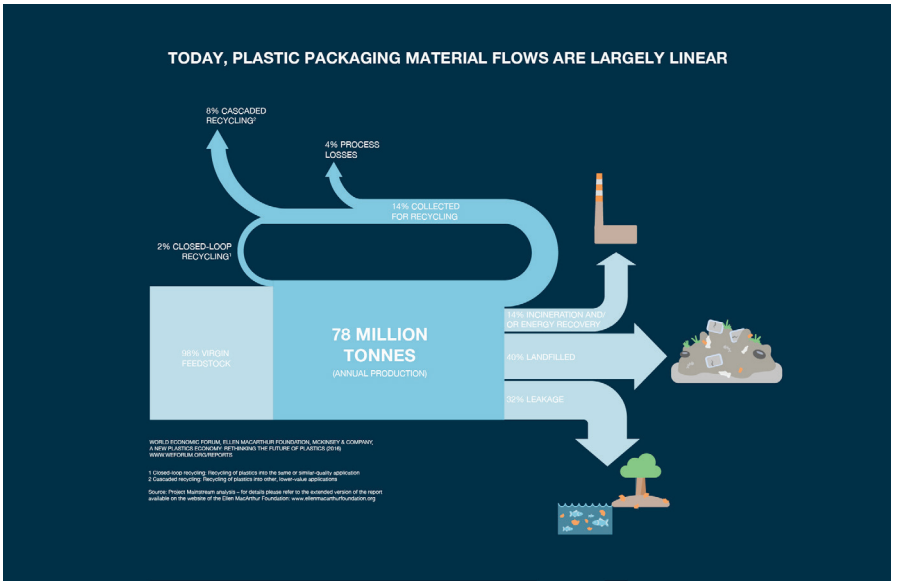


Diagramm 2.1-1 Linear Economy

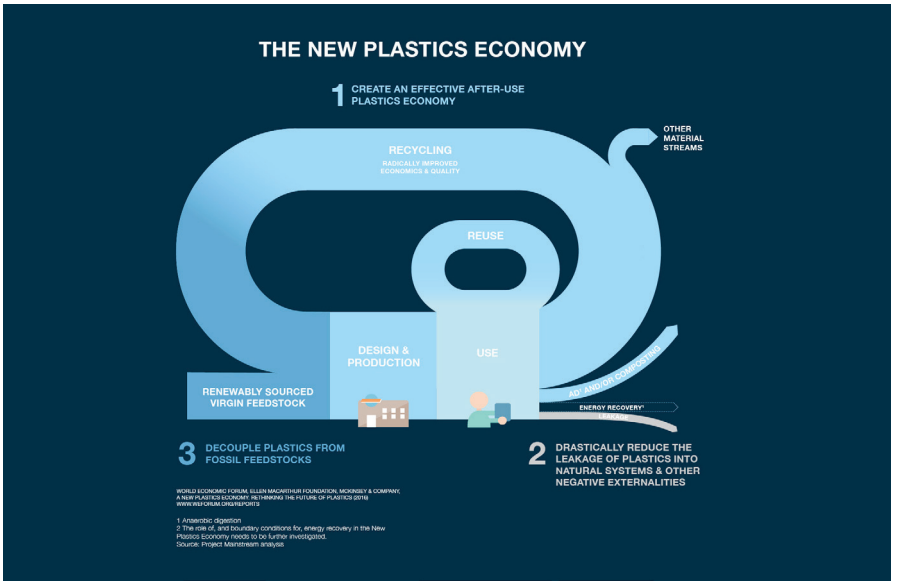


Diagramm 2.1-2 Circular Economy

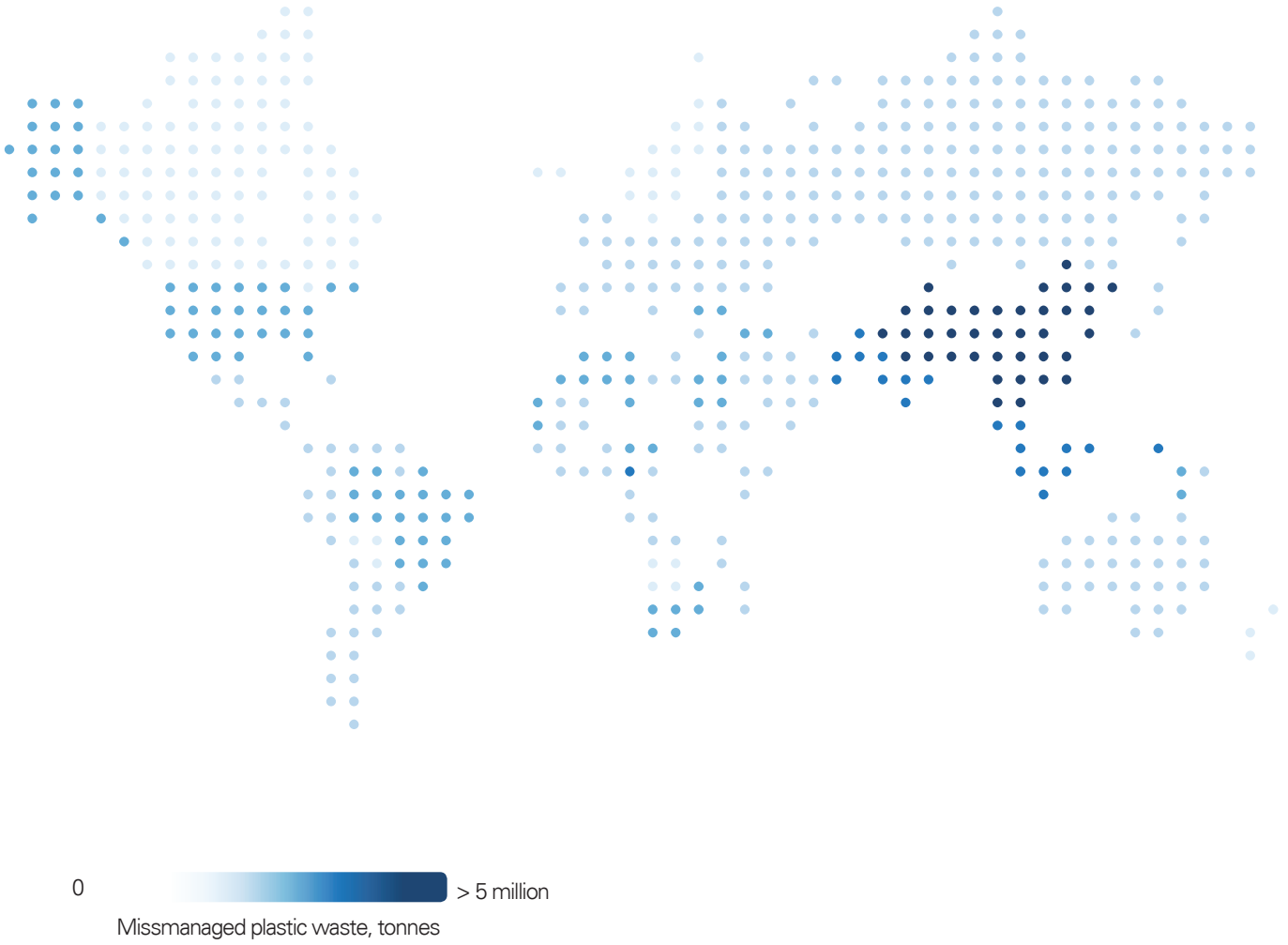
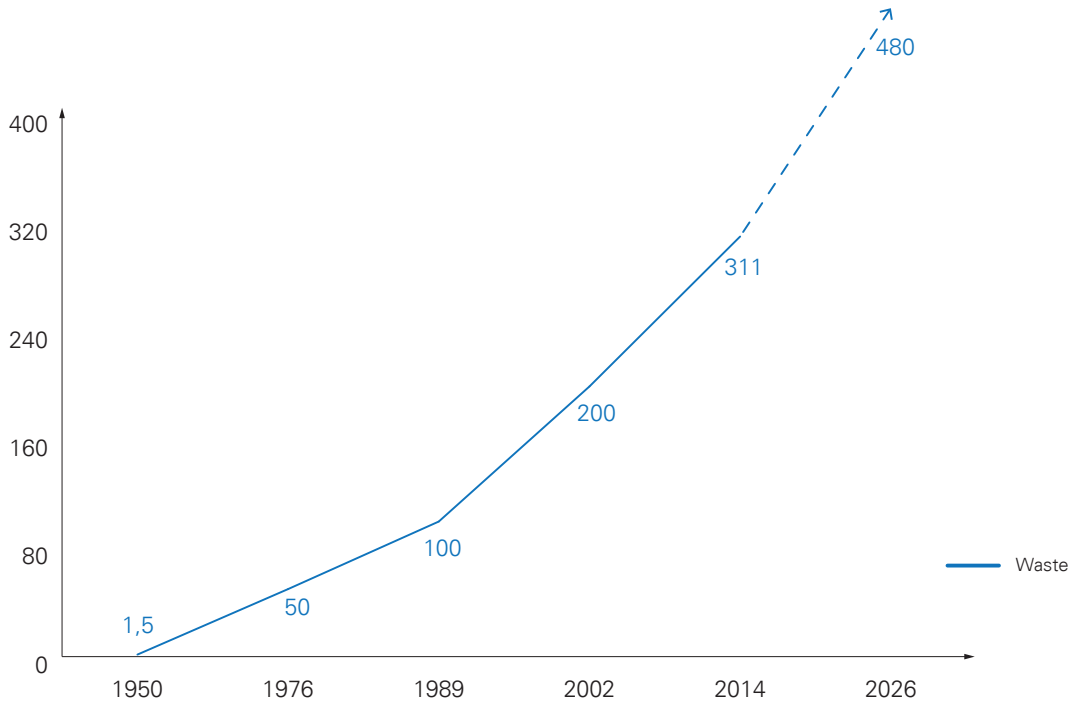


Diagramm 2.1-3 Nicht-recycleter Plastikmüll

Dass die weltweite Kunstoffproduktion immer größere Züge annimmt lässt sich an Zahlen und Daten festmachen. So steigt die Jährliche Produktion von Plastik (Abbildung 2.1-4) seit den 1950er Jahren mit zunehmender Geschwindigkeit. Zugleich nimmt auch die Zahl des Plastikmülls zu welcher aus dem System fällt und keinem recycling zugefürt werden kann (Abbildung 2.1-3). Dieser Müll endet dann häufig in den Weltmeeren und an den Stränden. Hierbei geht eine enorme menge an Rohstoffen verloren.



Global plasic production, million tons

Diagramm 2.1-4 Globale Kunstoffproduktion

2.2 robotik in der architektur

„Ob Bauroboter, Industrie 4.0, Drohnen oder intelligente Sensoren – die Robotik hält unweigerlich Einzug in unser Leben, den Alltag und auch den Architekturherstellungsprozess. Digitale Entwurfs- und Fertigungsmethoden verändern die konstruktiven und gestalterischen Möglichkeiten zunehmend.“ (Bettina Sigmund in Detail)

Roboter in der Architektur werden heute schon in manchen Bereichen eingesetzt. Häufig werden Roboter aus der Industrie entlehnt, welche z.B. in der Automobilindustrie schon seit Jahrzehnten zum Einsatz kommen. Diese Roboter weisen eine hohe Präzision auf und können mit verschiedenen Werkzeugen ausgestattet werden. Das Feld der autonomen Robotik in der Architektur in das auch diese Arbeit fällt ist ein neues Forschungsfeld. Die Roboter werden hierbei nicht mehr nur als Werkzeuge betrachtet sondern bilden eine neue „Companion species“ (Donna Haraway, 2003) die neben dem Menschen existiert und selbständig Probleme angeht und Lösungen findet.

Detail - Boom der Robotik, 2017
Donna Haraway, 2003

2.3 verortung

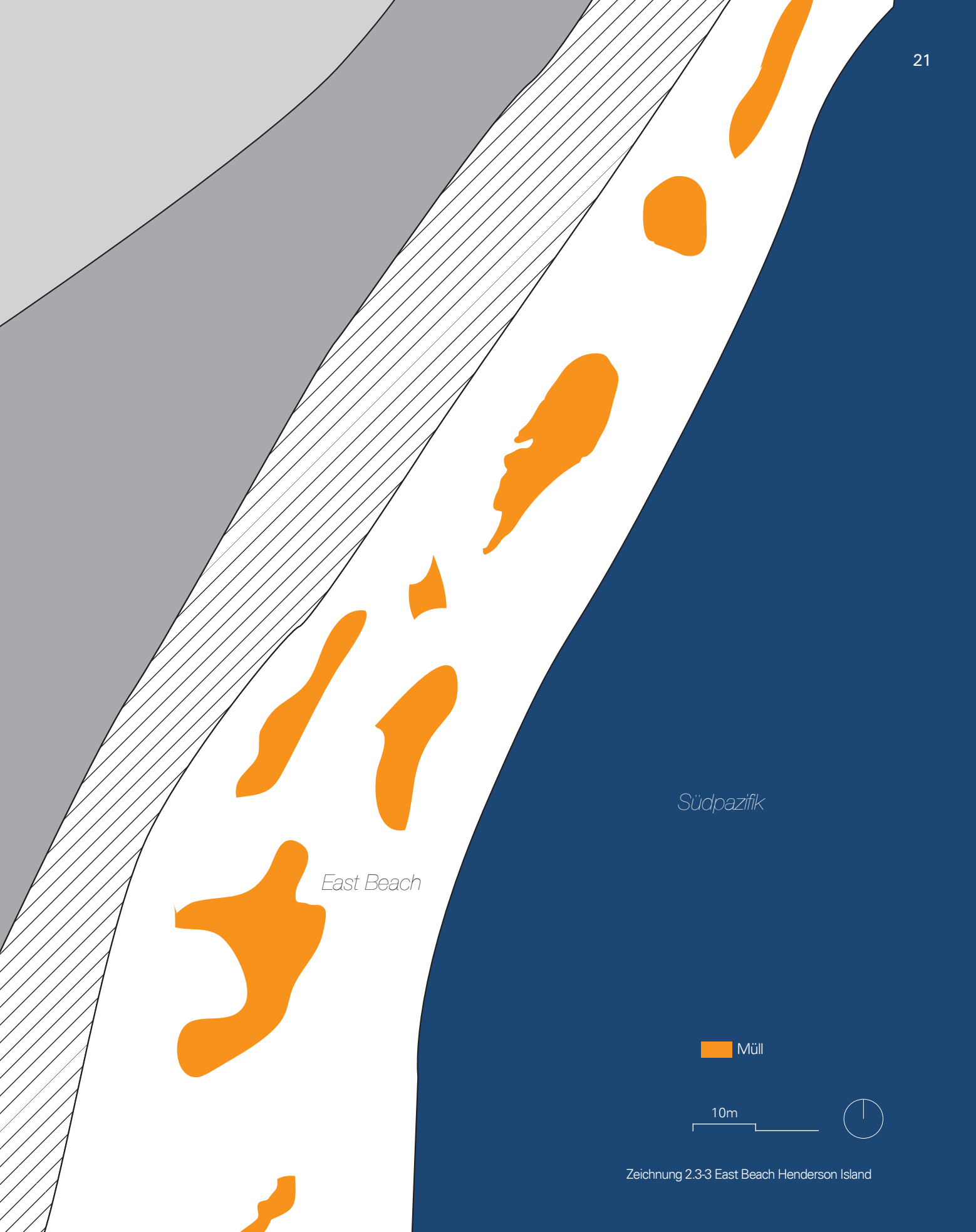
Henderson Island unbewohnte Pazifikinsel, die Tausende von Kilometern von der Zivilisation entfernt liegt, ist zum am meisten verschmutzten Ort der Erde geworden, als tausende von Plastikmüll an ihren Ufern gespült wurden. Henderson Island, ein Korallenatoll im Südpazifik, ist nur 37,5 Quadratkilometer groß und die nächsten Städte sind 4.800 Kilometer entfernt.
Die Insel, ein Territorium des Vereinigten Königreichs, ist ein Zufluchtsort für eine Vielzahl von bedrohten Arten, einschließlich der Henderson Petrel und Henderson Crake, und seine Strände sind ein Nistplatz für die vom Aussterben bedrohte grüne Schildkröte.
Die Insel soll als Szenario für das vorliegende Projekt dienen und auf die Thematik aufmerksam machen.

UNESCO - Henderson Island

Zeichnung 2.3-1 Weltkarte mit „Gyren“

Zeichnung 2.3-2 Henderson Island

Für das Szenario wurde der „East Beach“ an der Ostküste der Insel ausgewählt. Dieser Strand wurde exemplarisch für die Analyse ausgewählt.



Zeichnung 2.3-3 East Beach Henderson Island

3 ziele der arbeit

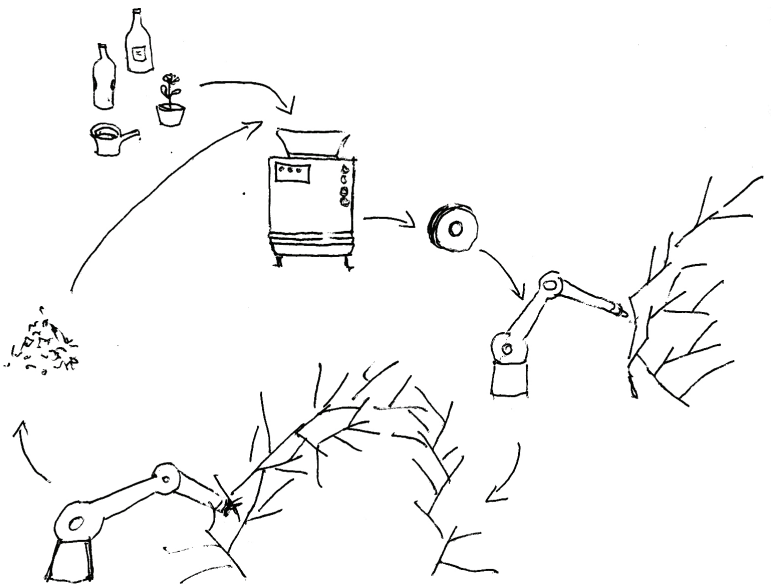
Die vorliegende Arbeit soll Wege aufzeigen wie Robotik in der Architektur eingesetzt werden kann. In diesem Szenario werden autonome Robotereinheiten entwickelt, die an verschmutzten Stränden ausgesetzt werden. Über einen mehrjährigen Zeitraum werden die müllverschmutzten Strände von den Robotern aufgeräumt indem der Müll wieder einem Recyclingkreislauf zugeführt wird und aus dem recycelten Material neue, nutzbare Strukturen erschaffen werden. Denkbare Anwendungen reichen von Sortierkörben für nicht recyclebares Material über Liegeobjekte für Strandbesucher bis hin zu Mikroarchitektur die z.B. Schutz vor Sonne und Wind bieten.
Die autonomen Robotereinheiten verfügen über identische Fähigkeiten und sind keiner hierarchischen Struktur unterlegen. Damit kann auch der Ausfall einer Einheit kompensiert werden. Die Maschinen kooperieren miteinander und teilen die von ihnen gesammelten Informationen.
Im nächsten Schritt wird der weitere Forschungsschwerpunkt auf die Entwicklung eines exemplarischen Roboters als „Werkzeugträger" gelegt.
Mit der Verortung und der Aufgabenstellung der Roboter soll zudem Awareness hinsichtlich der Plastikmüllproblematik geschaffen werden.

4 methodik

Da das Projekt verschiedene Methoden vereint werden im Folgenden essentielle Methoden und Vorgehensweisen erklärt. Dies umfasst von der Formfindung bis hin zur Programmierung verschiedene Bereiche von Digital Fabrication, Swarm Robotics in der Architektur und Artificial Intelligence.

4.1 einstieg

Am Anfang der Arbeit wurden Möglichkeiten gesucht wie Formen generiert werden können, die mit Hilfe von Robotern erstellt werden können. Ein erstes Konzept (Abbildung 4.1-1) beschreibt eine „Recycling-Architektur" bei der Roboter Strukturen erstellen, diese umbauen und auch wieder recyclen können.
Im Bereich der Form wurden Experimente mit Tensile Grid Strukturen gemacht.



Zeichnung 4.1-1 Schema Recyclingroboter

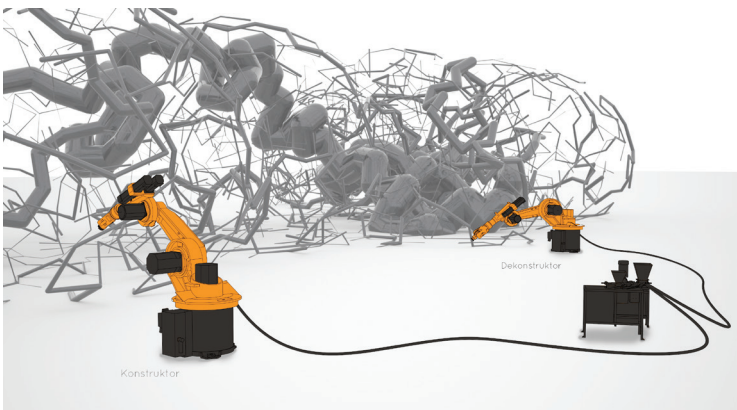
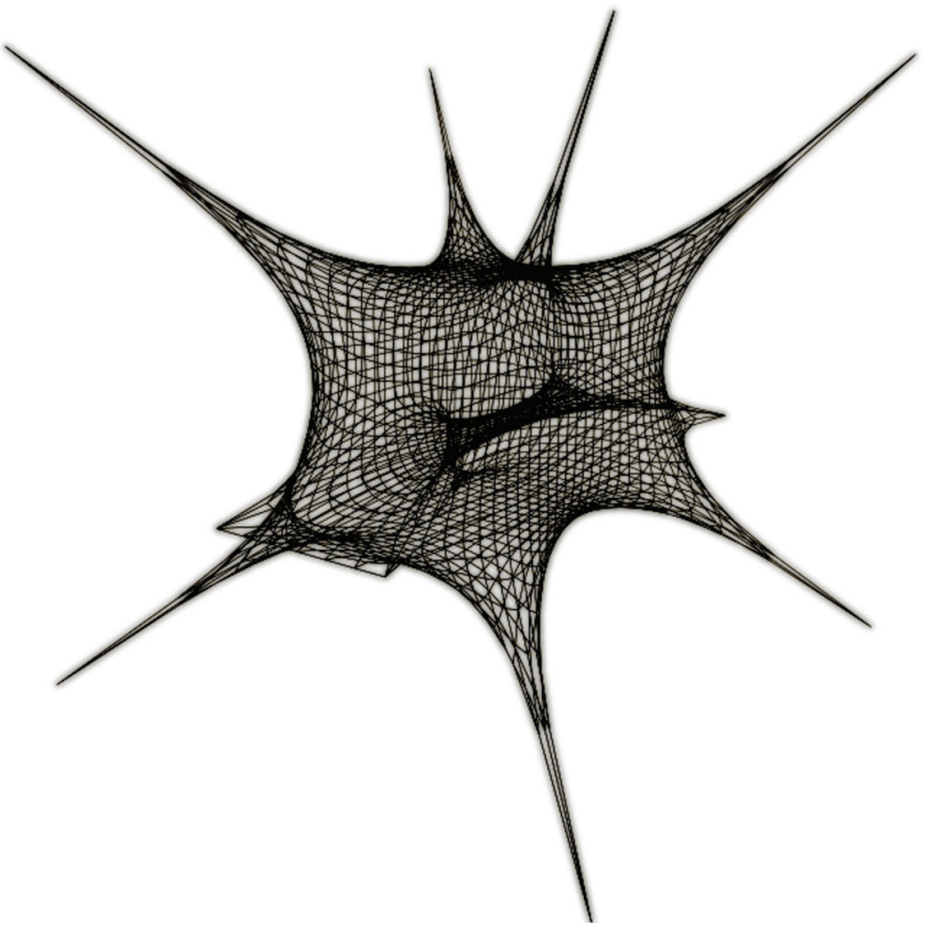
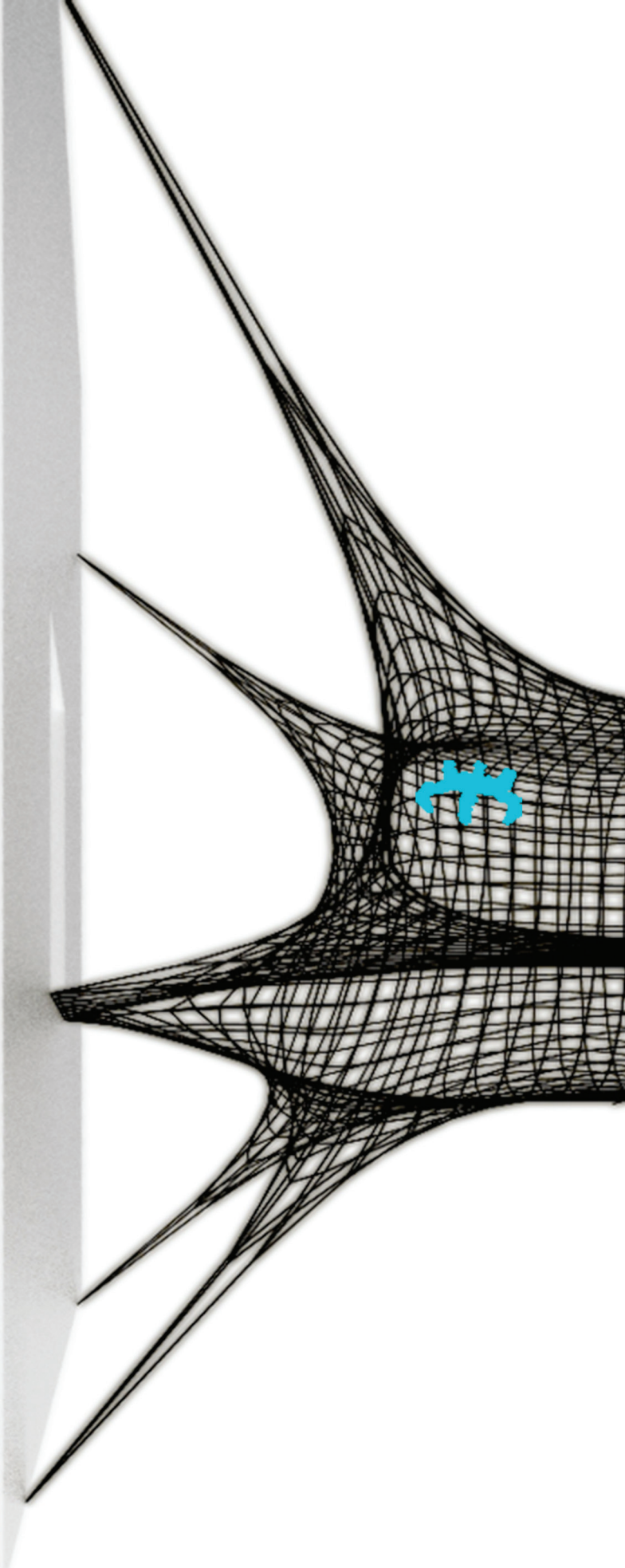


Abbildung 4.1-1 Schema Konstruktor/Destruktor



Zeichnung 4.1-2 Tensile Grid



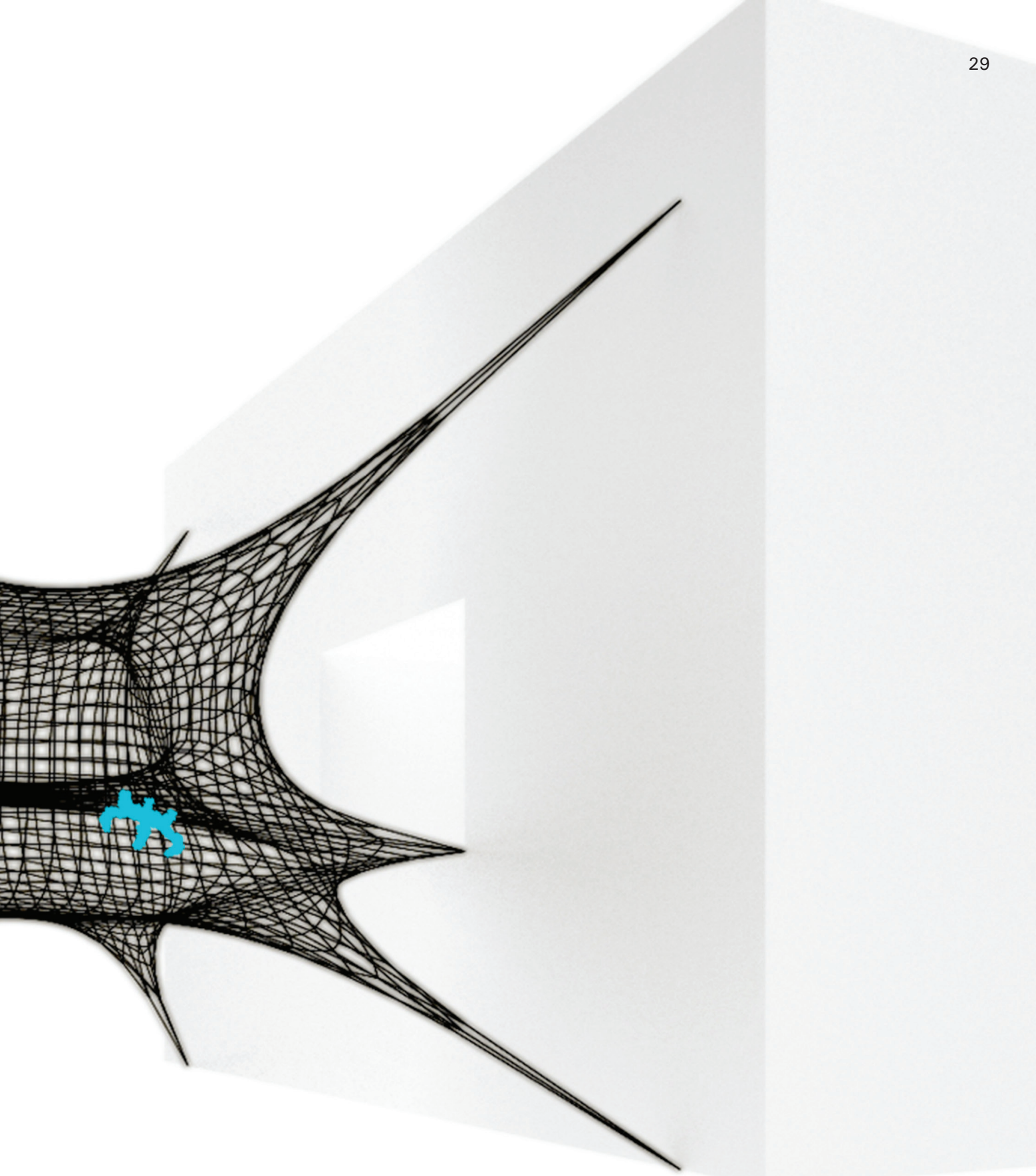


Abbildung 4.1-1 Tensile Grid mit Agenten

4.2 exkurs: victimless crime

Bei einem mehrtägigen Workshop in Bratislava (Team: Christoph Müller, Michael Schultes, Jan Pernecky, Thomas Tholt, Stefan Pirintchev, Anel Bucan, Konrad Zellner und Christoph Kolbeck) im Dezember 2017 wurden mithilfe eines KUKA Roboters untersucht wie Strukturen aus dem Rechner in die Realität übersetzt werden können. Hierbei konnten wichtige Erkenntnisse im Bereich der Digital Fabrication gemacht werden, welche die Arbeit entscheidend beeinflusst haben. Zum einen wurden Arbeitsabläufe mit dem Roboter und zum anderen auch das Tooling untersucht.

Abbildung 4.2-1 Experiment zur Materialselbstorganisation

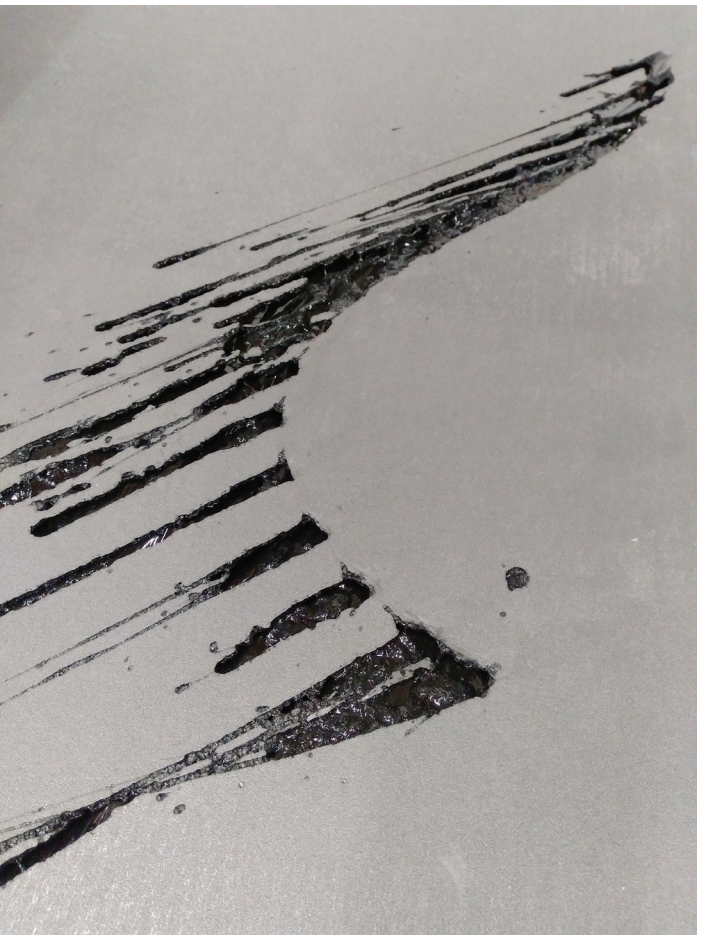


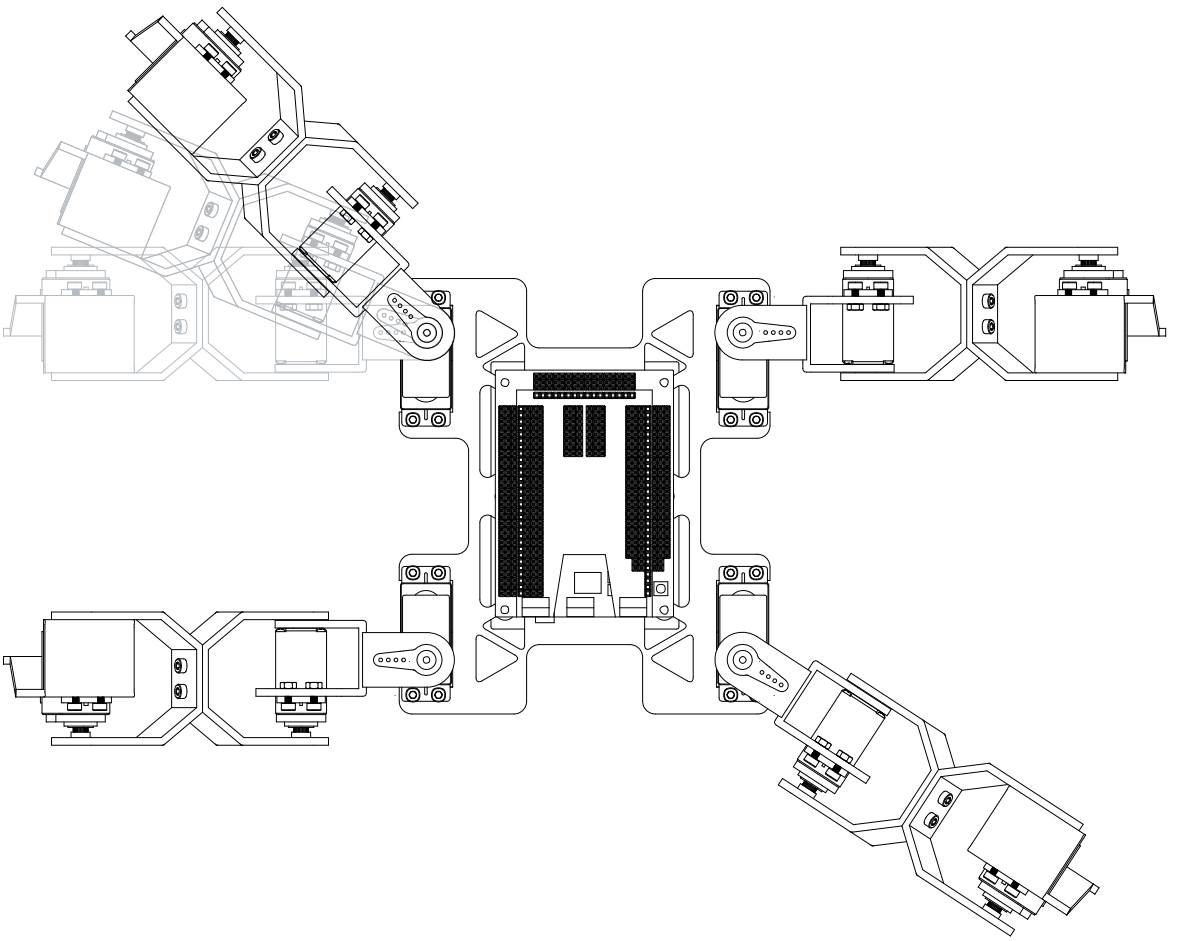
Abbildung 4.2-2 Experiment zu Werkzeugpfad

Abbildung 4.2-3 Pneu

4.3 quadropoden

Lauffähige Roboter können auf allen Arten von Oberflächen navigieren die für Roboter mit Rädern nicht zugänglich sind.
Ein großer Vorteil von lauffähigen Robotern ist, dass sie nicht auf eine bestimmte Infrastruktur wie glatte Oberflächen, Straßen, Schienen usw. angewiesen sind und daher auch in rauem, felsigen, sandigen oder unbekanntem Gelände zum Einsatz kommen können. Entsprechend sind lauffähige Roboter auch deutlich flexibler wenn es um das Bewältigen eines Hindernisses geht, sie können darüber springen oder es übertreten.
Dagegen müssen Roboter mit Rädern einen anderen Weg nehmen, wenn das darüber fahren nicht möglich ist.
Die Besonderheit an vierbeinigen Robotern (Quadropoden) ist, dass sie in der Lage sind stabil auf drei Beinen zu stehen (drei beliebige Punkte im Raum bilden immer eine Fläche) und das vierte als Werkzeug verwenden können.

Robot Platform, 2013



Zeichnung 4.3-1 Schema Bewegung Quadropde

4.4 inverse kinematik

Inverse Kinematik ist ein Konzept welches ermöglicht einen gegebenen Punkt im Raum mittels mehrerer Gelenke zu erreichen. Um das zu veranschaulichen stelle man sich ein Glas Wasser vor, welches man anheben möchte. Diese Verbindung kann beschrieben werden als Position „Hand" nach Position „Glas". Dieser Zusammenhang ist allerdings nicht trivial, da das Gehirn dafür Sorge tragen muss welche Winkel die einzelnen Gelenke des Arms (Schultergelenk, Ellbogengelenk, Handgelenk) annehmen müssen. Hierbei gibt es weit mehr als nur eine Lösung.

Die Berechnung erfolgt mittels trigonometrischer Formeln, welche die notwendigen Winkel an den Gelenken im Bezug zum gewünschten Endpunkt und dem Zentrum der Masse haben. Hierzu zerlegt man das dreidimensionale Problem in zwei zweidimensionale Probleme (Oscar Liang) (s. Diagramm 4.4-1). Dadurch erhält man eine Formel welche später im Skript zum Einsatz kommt.

Dieses Konzept kann nun genutzt werden um Roboter mit mehreren Beinen und DOF (Dimensions of freedom) in Bewegung zu versetzen und verschiedene Aufgaben mit den Beinen zu erfüllen.

Zerlegen des Problems von einem 3D in zwei 2D Probleme um α, β und γ zu lösen.
γ lässt sich aus dem Diagramm entnehmen (Draufsicht). Die restlichen beiden Winkel befinden sich nun in der selben Ebene (Seitenansicht).

$$\frac{x}{y} = \tan(\gamma)$$

$$\rightarrow \gamma = \tan^{-1}\left(\frac{x}{y}\right)$$

Um α zu lösen wird es in in α_1 und α_2 aufgeteilt.
α_1 kann über die Länge L ermittelt werden:

$$L = \sqrt[2]{z_{offset}^2 + (L1 - coxa)^2}$$

$$\alpha_1 = \cos^{-1}\left(\frac{Z_{offset}}{L}\right)$$

α_2 and β werden mithilfe des Cosinus ermittelt:

$$a^2 = b^2 + c^2 - 2bc\ \cos A \qquad \cos A = \frac{b^2 + c^2 - a^2}{2bc}$$
$$b^2 = a^2 + c^2 - 2ac\ \cos B \qquad \cos B = \frac{a^2 + c^2 - b^2}{2ac}$$
$$c^2 = b^2 + a^2 - 2ab\cos C \qquad \cos C = \frac{a^2 + b^2 - c^2}{2ab}$$

Anhand der Formel und den drei bekannten Seiten des Dreiecks können alle Winkel des selbigen definiert werden.

Damit ist α_2 gelöst:

$$Tibia^2 = Femur^2 + L^2 - 2(Femur)(L)\cos(\alpha_2)$$

$$\rightarrow \alpha_2 = \cos^{-1}\frac{Tibia^2 - Femur^2 - L^2}{-2(Femur)(L)}$$

Damit ist α:

$$\alpha = \alpha_1 + \alpha_2$$

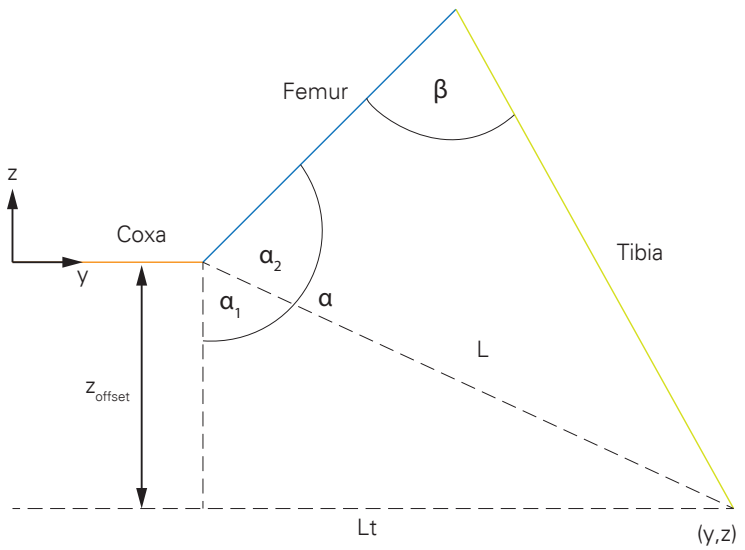
$$\alpha = \cos^{-1}\left(\frac{Z_{offset}}{L}\right) + \cos^{-1}\frac{Tibia^2 - Femur^2 - L^2}{-2(Femur)(L)}$$

Und schlussendlich β:

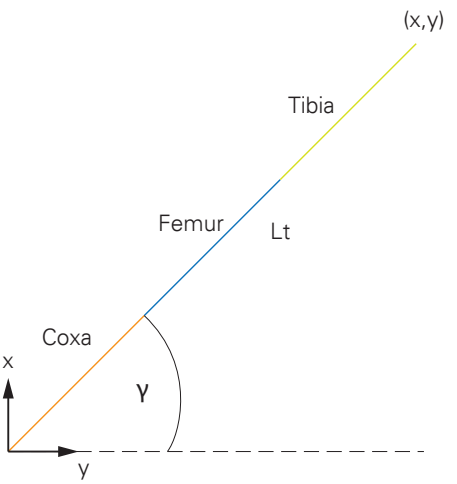
$$L^2 = Tibia^2 + Femur^2 - 2(Tibia)(Femur)\cos(\beta)$$

$$\rightarrow \beta = \cos^{-1}\frac{L^2 - Tibia^2 - Femur^2}{-2(Tibia)(Femur)}$$

Oscar Liang, 2012



Seitenansicht



Draufsicht

Diagramm 4.4-1 Schema Inverskinematik

4.5 roboter funktionsschema

Ein Agent setzt sich aus verschiedenen Elementen zusammen, welche miteinander verbunden die Funktionalität gewährleisten. Hierbei gibt es zwei Hauptelemente: den Microprozessor („Großhirn“) und den Microcontroller („Kleinhirn“).
Der Mikroprozessor ist für die Berechnung und Erzeugung der Strukturgeometrie zuständig. Er berechnet aus den generierten Strukturen mit Hilfe eines Inverskinematiksystem (s. Inverse Kinematik) die Eingaben für den Microontroller. Diese werden mittels einer seriellen Schnittstelle übermittelt. Darüber hinaus kann der Microprozessor aufwendigere Eingaben wie beispielsweise das Lesen und Auswerten von Bildern einer angeschlossenen Kamera übernehmen. Der Microcontroller ist für die Koordination der Servomotoren und das Auslesen und Verarbeiten von Sensordaten auf niedrigem Niveau zuständig. Hierunter fallen Sensoren für die Erkennung von Hindernissen (Obstacle avoidance) sowie das Erkennen von Objekten. Darüberhinaus steuert er den Printercontroller welcher die Parameter für den Druckkopf (Vorschubgeschwindigkeit, Temperatur) steuert. Die Kombination von Microprozessor und Microcontroller ermöglicht eine schnelle Reaktion auf die Umwelt sowie die präzise Umsetzung komplexer Aufgaben.

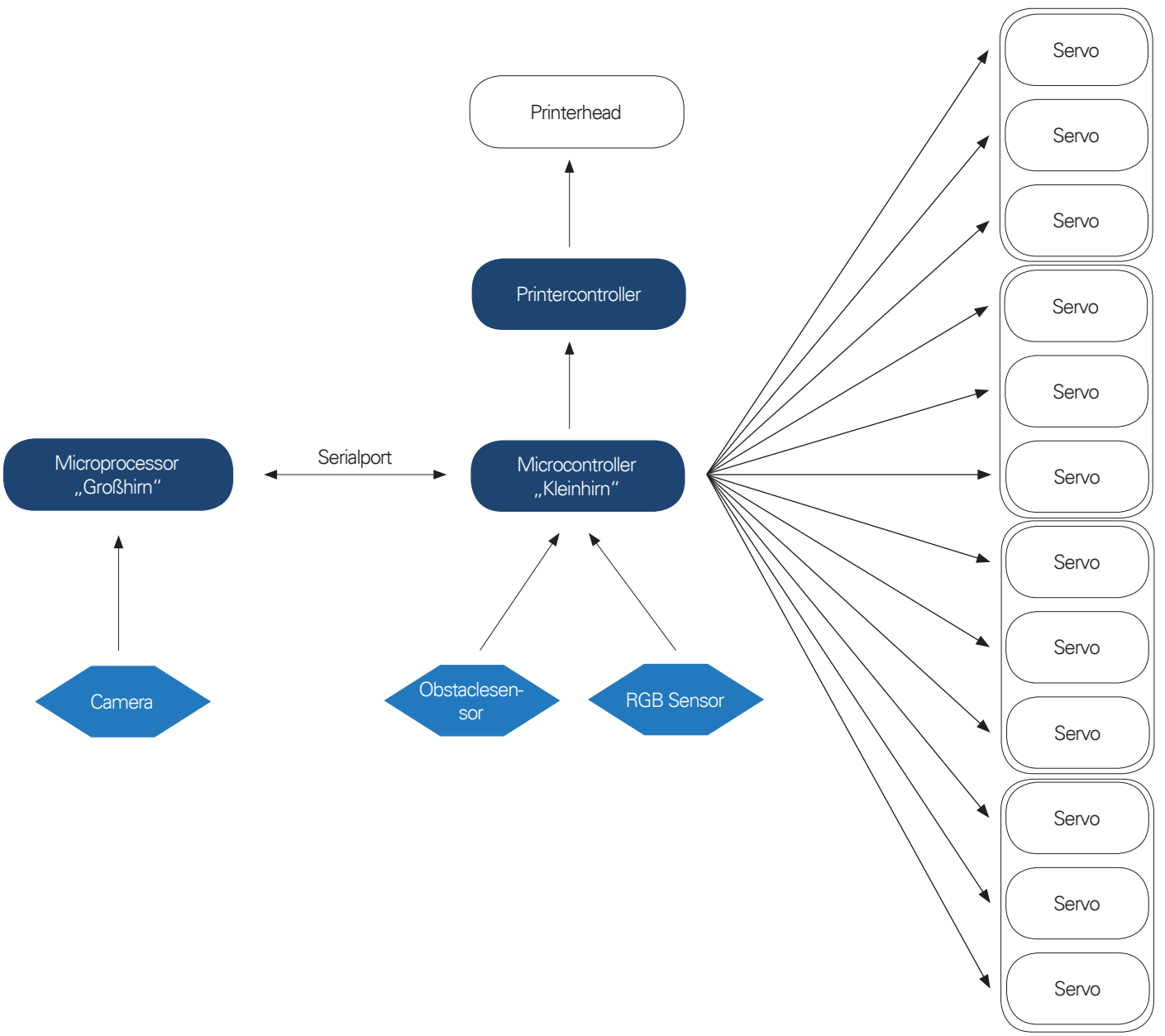


Diagramm 4.5-1 Schema Roboterschaltplan

4.6 software controller

Damit Microprozessor und Controller miteinander kommunizieren können wurde im Rahmen der vorliegenden Arbeit eine Software-Schnittstelle adaptiert und eine Sprache die auf einfachen Zahlencodes basiert entwickelt. Zunächst wird die Programmierung des Microcontrollers betrachtet. Der Controller fragt kontinuierlich die serielle Schnittstelle ab und führt, sobald eine Übertragung aktiv ist, das nebenstehende Skript aus. Hiermit wird die Kontrolle an den Microprozessor übergeben und spezifische Aufgaben wie das Drucken von Strukturen gelöst. Ist die Aufgabe beendet wird die Kontrolle wieder an den Controller übergeben.

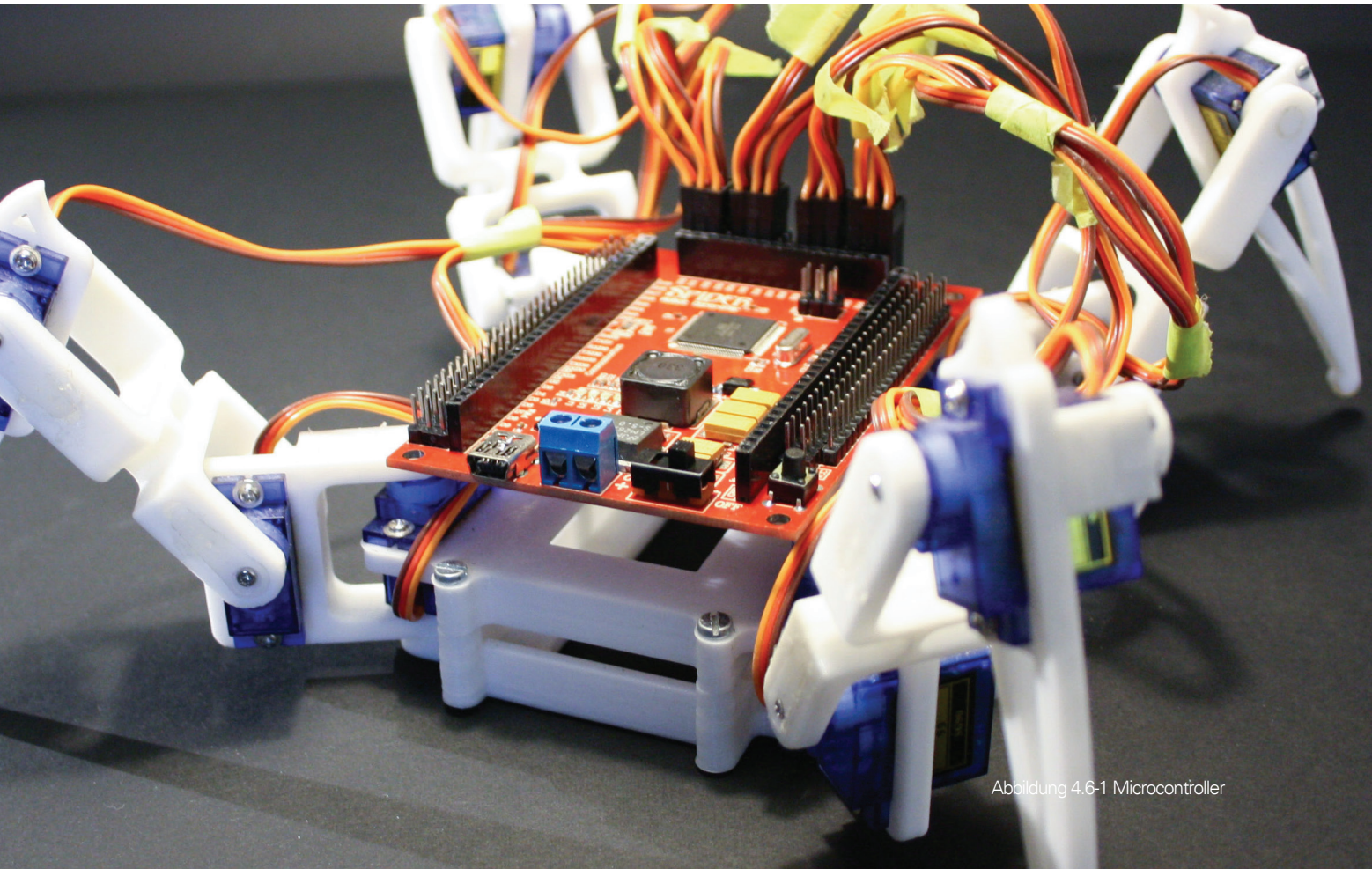


Abbildung 4.6-1 Microcontroller

```
void loop(void) {

  if (Serial.available() > 0) {

    int receiving = 1;

    if (receiving == 1) {

      int action = Serial.parseInt();

      // to print
      if (action == 1) {
        printer();
      }

      // to stop flexitimer
      if (action == 2) {
        FlexiTimer2::stop();
        delay(15);
      }

      // to stand
      if (action == 5) {
        stand();
        delay(2000);
      }

      // to sit
      if (action == 6) {
        sit();
        delay(100);
      }

      // to default position
      if (action == 7) {
        default_position();
        delay(2000);
      }


      // to recieve from blender
      if (action == 8) {
        // empfangen

        blender_control_servo_1 = Serial.parseInt();
        blender_control_servo_2 = Serial.parseInt();
        blender_control_servo_3 = Serial.parseInt();

        // position anfahren

        servo[1][2].write(blender_control_servo_1);
        delay(15);
        servo[1][0].write(blender_control_servo_2);
        delay(15);
        servo[1][1].write(blender_control_servo_3);
        delay(15);


      }

      if (action == 9) {
        FlexiTimer2::start();
        delay(15);
        default_position();
        delay(2000);
        stand();
        delay(2000);
        step_forward(3);
        turn_right(5);
        receiving = 0;
        delay(2000);


      }
    }
  }

  else {
    explore();
  }
}
```

Skript 4.6-1 Programmierung Microcontroller, Sprache: C++

4.7 software prozessor

Auf der anderen Seite der Schnittstelle steht die Programmierung des Microprozessors. Im nebenstehenden Skript werden die generierten Strukturen in Daten übersetzt die für den Microcontroller verarbeitbar sind und an selbigen in Echtzeit gesendet werden. Das „Großhirn“ übernimmt an dieser Stelle die Kontrolle über den Agenten.

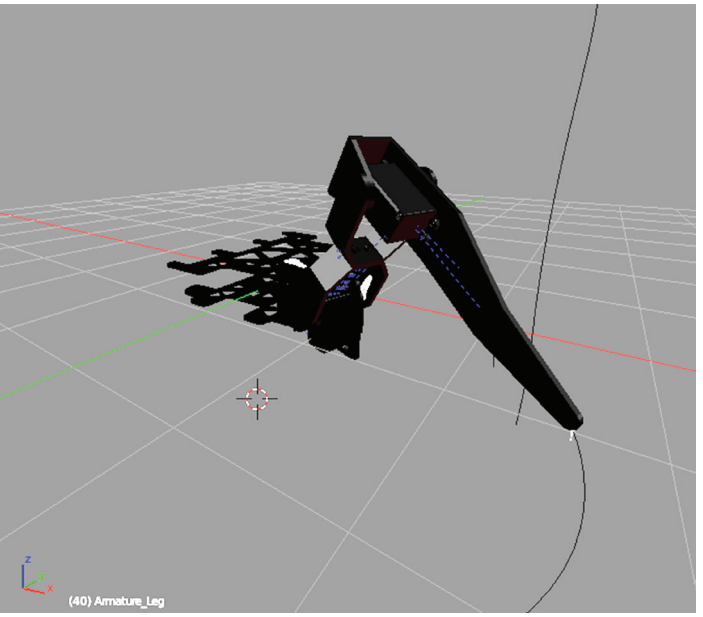


Abbildung 4.7-1 Datenberechnung in Blender

```
import bpy
import math
from math import degrees
import mathutils
import serial
import time

# Setup für serial

Port = „/dev/cu.HC-06-DevB“
ser = serial.Serial(Port,9600,timeout=1)
printpath = bpy.context.object.pose.
bones[„Target“].constraints[„Follow Path“].
target

# Roboterarmdefinition

arm = bpy.data.objects[„Leg“]

pb1 = arm.pose.bones.get(„Hip“)
pb2 = arm.pose.bones.get(„Coxa“)
pb3 = arm.pose.bones.get(„Femur“)
pb4 = arm.pose.bones.get(„Tibia“)
pb5 = arm.pose.bones.get(„Target“)

# Funktionen des Roboters

def stand():
    print(„stand“)
    ser.write(b“5“)
    ser.write(b“;“)

def sit():
    print(„sit“)
    ser.write(b“6“)
    ser.write(b“;“)

def default_position():
    print(„default position“)
    ser.write(b“7“)
    ser.write(b“;“)
def printer():
```

```
    print(„printer switch“)
    ser.write(b“1“)
    ser.write(b“;“)

def initiate():         #Flexitimer Stop
    print(„initiate“)
    ser.write(b“2“)
    ser.write(b“;“)

def reload():
    print(„reload“)
    ser.write(b“9“)
    ser.write(b“;“)

def path():
    for frame in range(100):

        # read angles
        v1 = pb1.head - pb1.tail
        v2 = pb2.head - pb2.tail
        v3 = pb3.head - pb3.tail
        v4 = pb4.head - pb4.tail

        if pb1 and pb2:
            # berechnet den Winkel wie er in
blender ist
            servo_angle_1 = int(degrees(v1.
angle(v2)))
            servo_angle_2 = int(degrees(v2.
angle(v3))+90)
            servo_angle_3 = int(degrees(v3.
angle(v4)))

            print(„digital angles    frame“,
frame, „   angle_1“, servo_angle_1, „   ang-
le_2“, servo_angle_2, „   angle_3“, servo_ang-
le_3)
            print(„physical angles   frame“,
frame, „   angle_1“, servo_angle_1, „   ang-
le_2“, servo_angle_2, „   angle_3“, servo_ang-
le_3)
            servo_angle_1 = bytes(str(servo_ang-
le_1).encode())
            servo_angle_2 = bytes(str(servo_ang-
le_2).encode())
            servo_angle_3 = bytes(str(servo_ang-
le_3).encode())

            ser.write(b“8“)
            ser.write(b“;“)

            ser.write(servo_angle_1)
            ser.write(b“;“)

            ser.write(servo_angle_2)
            ser.write(b“;“)

            ser.write(servo_angle_3)
            ser.write(b“;“)

            time.sleep(0.1)

        else:
            print(„Invalid bone name(s)“)


        bpy.context.scene.frame_set(frame)
```

Skript 4.7-1 Programmierung Microprozessor Sprache: Python

4.8 ablaufdiagramm

Die Agenten folgen nachdem sie ausgesetzt wurden einem im Skript festgelegten Ablaufschema (Diagramm 4.8-1). Nach dem Einschalten führt der Microcontroller einen Initialisierungsprozess durch, lädt alle notwendigen Prozesse und kalibriert die Servomotoren in Ausgangsposition. Dannach wartet der Microcontroller bis eine Verbindung zum Microprozessor hergestellt ist und startet dann mit dem „Exploremode“ (s. Systematik 4.10), d.h. er erkundet eigenständig seine Umgebung. Gleichzeitig überprüft er immer die serielle Schnittstelle um Befehle vom Microprozesor entgegenzunehmen. Ist dies der Fall führt er solange die Befehle vom Microprozessor aus bis dieser das Ende der Aufgabe überträgt und nichts mehr über die Schnittstelle übertragen wird. Der Roboter geht so wieder in den „Eploremode“ zurück.

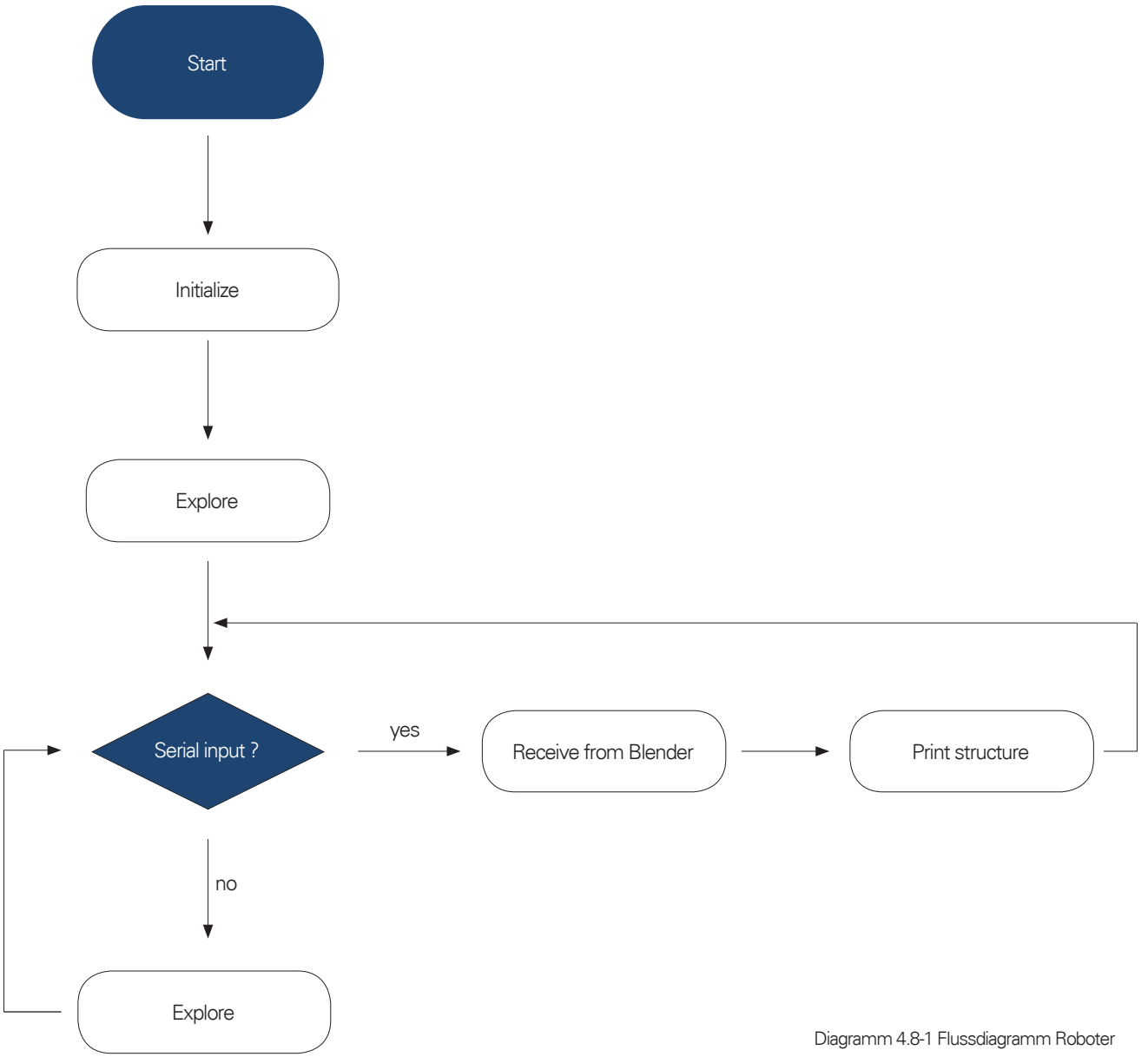
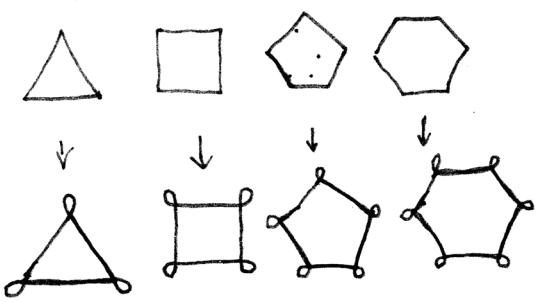


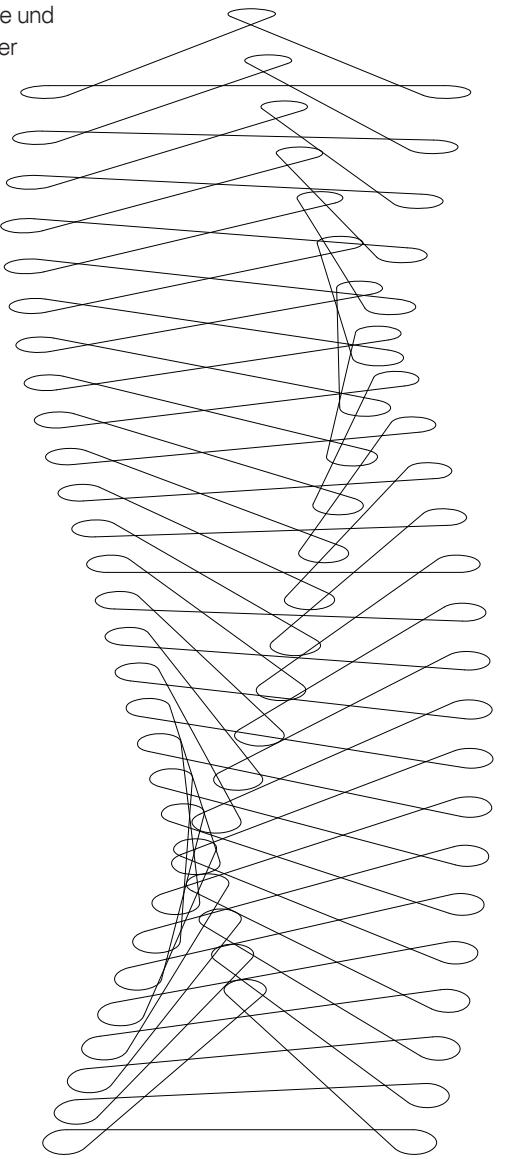
Diagramm 4.8-1 Flussdiagramm Roboter

4.9 formgeneration

Die Entwicklung der Form erfolgt ausgehend von Primitiven (Dreieck, Viereck, Fünfeck, etc.) auf reiner Script-basis mittels Python. Durch die Verwendung von primitiven Grundkörpern kann die Präzision festgestellt werden. Die Primitiven werden „geschlauft" (Zeichnung 4.9-1) und dann mittels Translation des Ortes, der Größe und der Ausrichtung verändert und so ein immer neuer Druckpfad erstellt. Die Agenten sind so in der Lage ausgehend von einfachster Geometrie in kurzer Zeit auch anspruchsvollere Strukturen zu erstellen. Die Strukturen werden auf Basis einer „ Bedürfnispyramide" generiert und können mithilfe eines machine learning mit zunehmender Präzision erstellt werden. Die Strukturen können durch die Beweglichkeit der Agenten miteinander kombiniert werden um so auch größere Objekte zu generieren.



Zeichnung 4.9-1 Schema „Verschlaufen"



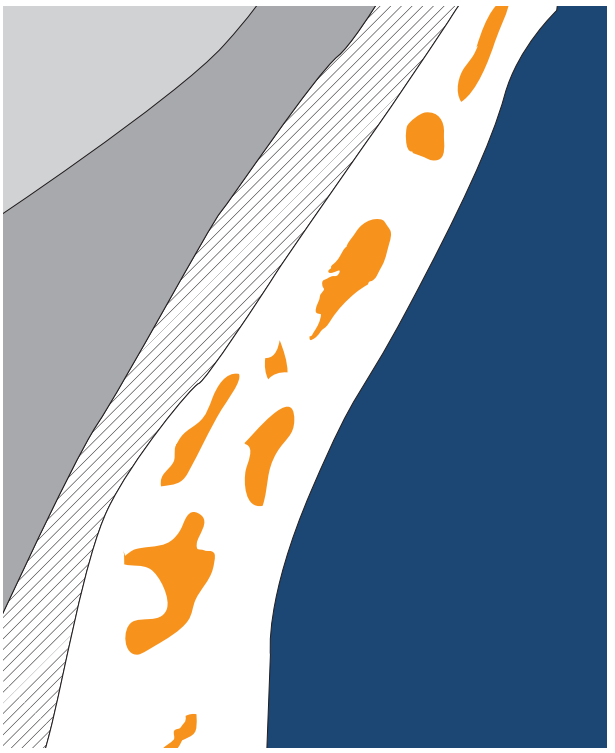
Zeichnung 4.9-2 Schema Formgeneration

```
def movepath_x():
    print(„move path x-axis“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.location.x += 0.02

def movepath_y():
    print(„move path y-axis“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.location.y += 0.02

def movepath_z():
    print(„move path z-axis“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.location.z += 0.005

def rotatepath_x():
    print(„rotate path x-axis“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.rotation_euler = (radians(0.1), 0, 0)

def rotatepath_y():
    print(„rotate path y-axis“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.rotation_euler = (0, radians(0.1), 0)

def rotatepath_z():
    print(„rotate path z-axis“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.rotation_euler = (0, 0, radians(0.1))

def scalepath_plus():
    print(„scale path“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.resize(value=(+0.1, +0.1, +0.1))

def scalepath_minus():
    print(„scale path“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.resize(value=(-0.1, -0.1, -0.1))

def resetpath():
    print(„reset path“)
    printpath = bpy.data.objects[„Cycloid.002“]
    printpath.location = (0.0,0.0,0.0)
    printpath.rotation_euler = (0, 0, 0)
```

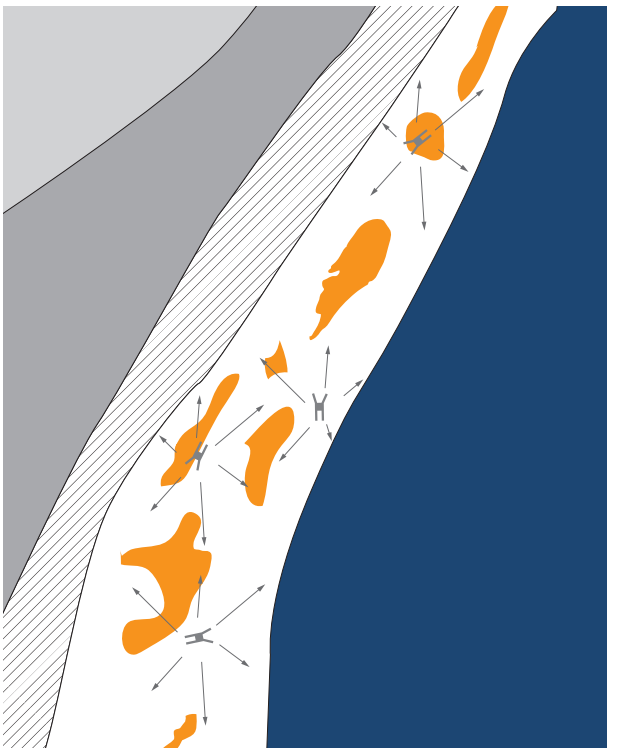
Skript 4.9-1 Programmierung Formgeneration, Sprache: Python

4.10 systematik

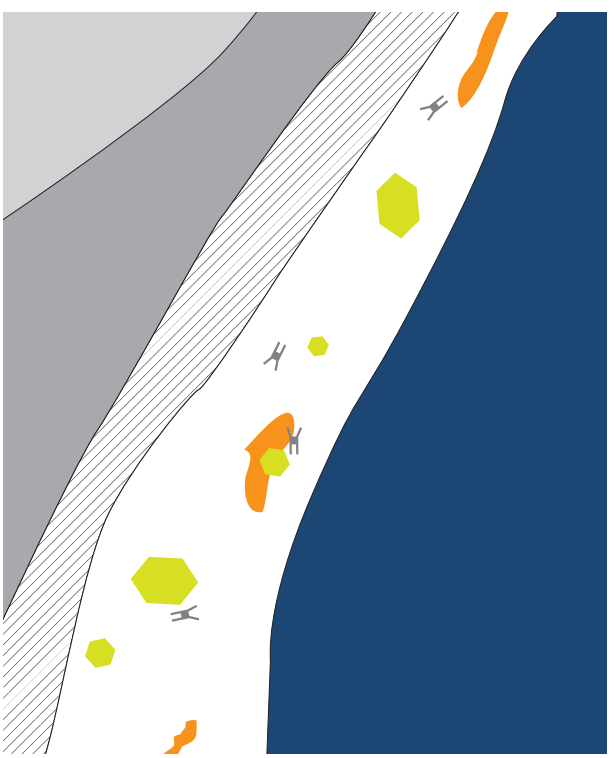
Nachdem die Robotereinheiten an einem verschmutztem Strand ausgesetzt wurden (Zeichnung 4.10-1), erkunden sie zunächst ihre Umgebung. Gleichzeitig erhalten sie über ihre Sensoren die notwendigen Umgebungsdaten, um den potentiellen Bauraum zu definieren (Zeichnung 4.10-2). Die gewonnenen Daten werden dezentral in einer Datenbank gespeichert und sind für alle Agenten nutzbar. Im nächsten Schritt beginnen die Agenten mit dem Sammeln und Kategorisieren von Material. Dabei werden zunächst Materialien gesammelt, die zum Bauen verwendet werden können. Wenn genug Baumaterial (s. Kunststoffarten 4.15) vorhanden ist, können „Sammelstellen" für die übrigen nicht verwertbaren Materialien gedruckt werden (Zeichnung 4.10-3).
Während zu Beginn primitive, für den Aufräumprozess notwendige Strukturen gedruckt werden, können in weiterer Folge immer komplexere Strukturen bis hin zu Architektur realisiert werden (Zeichnung 4.10-4).



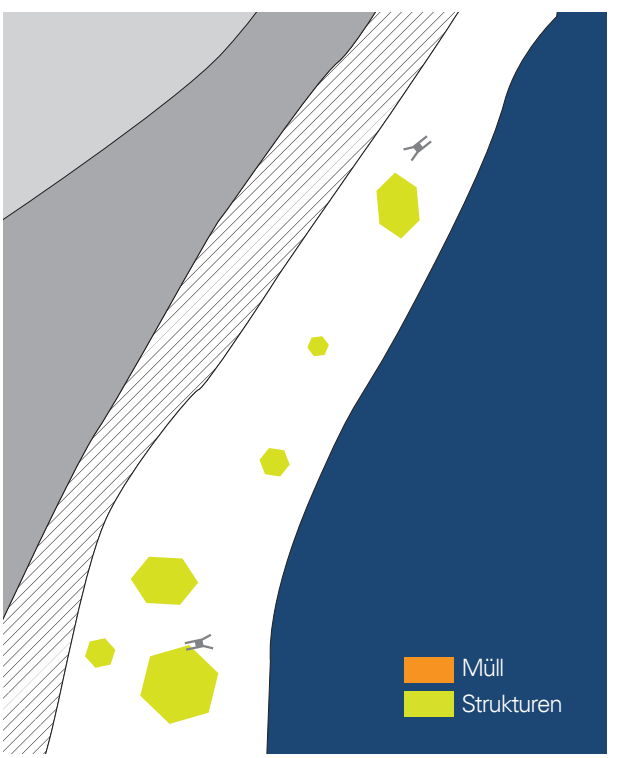
Zeichnung 4.10-1 Strand vor dem Aussetzen



Zeichnung 4.10-2 Exploremode



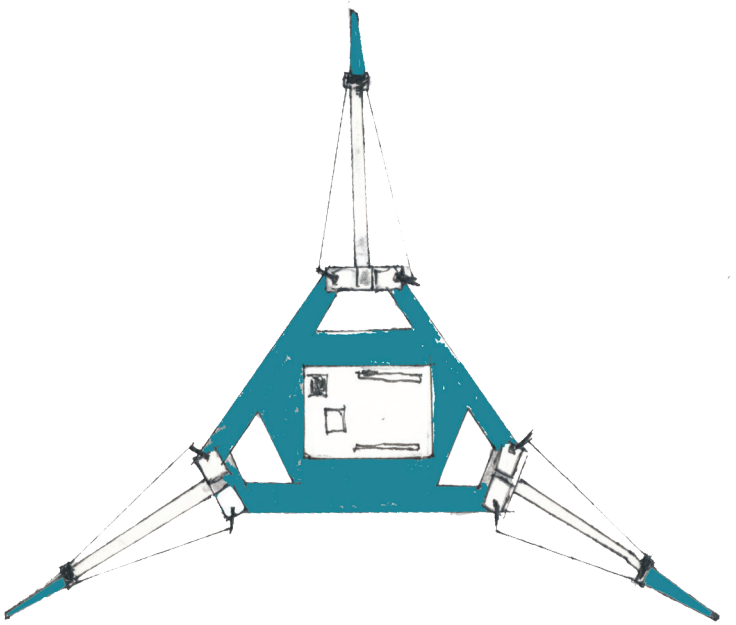
Zeichnung 4.10-3 Sammeln und Strukturen erstellen



Zeichnung 4.10-4 Strand mit Strukturen

4.11 roboterentwicklung

Die Basis für die Entwicklung des Agenten stellt ein Open-source Design von Regis Hsu da, welches für das Projekt von Grund auf neu entwickelt und an die Bedürnisse angepasst wurde. Das wichtigste Zitat aus seinem Design stellt die Art und Weise dar, wie er die Gelenkproblematik löst. In weiterentwickelter Form findet sich seine Idee auch im Prototypen der Agenten wieder.



Zeichnung 4.11-1 Tripod Robot

Regis Hsu, 2015

Abbildung 4.11-1 Spider Robot

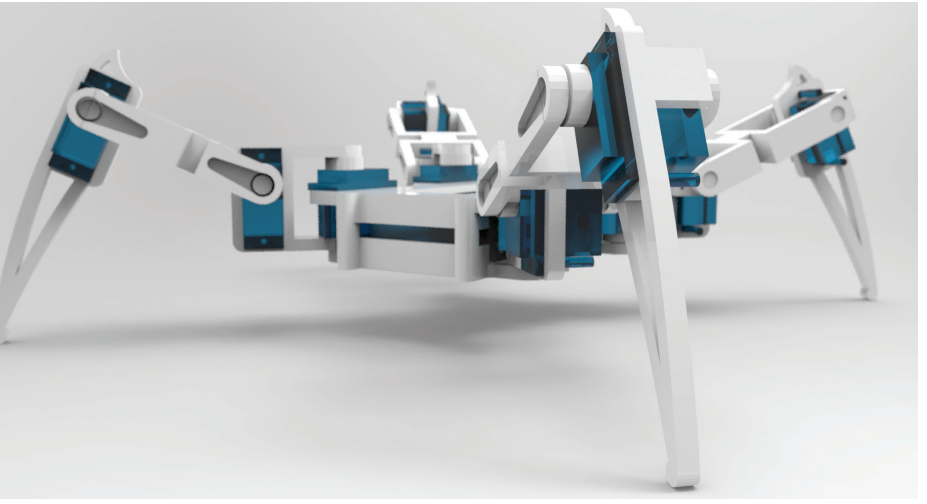


Abbildung 4.11-2 Spider Robot

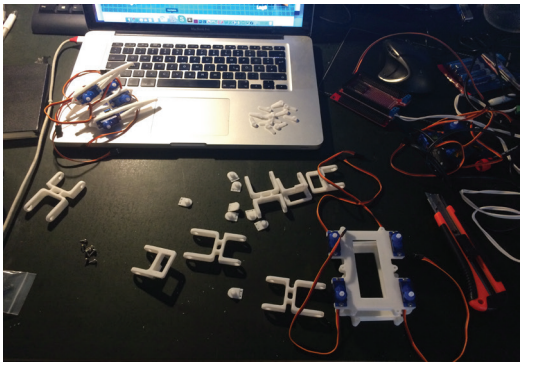


Abbildung 4.11-3 Zusammenbau

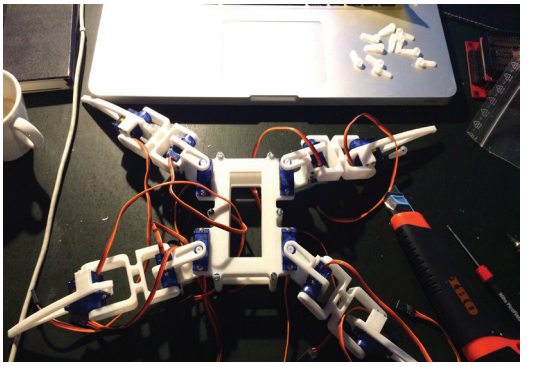


Abbildung 4.11-4 Zusammenbau

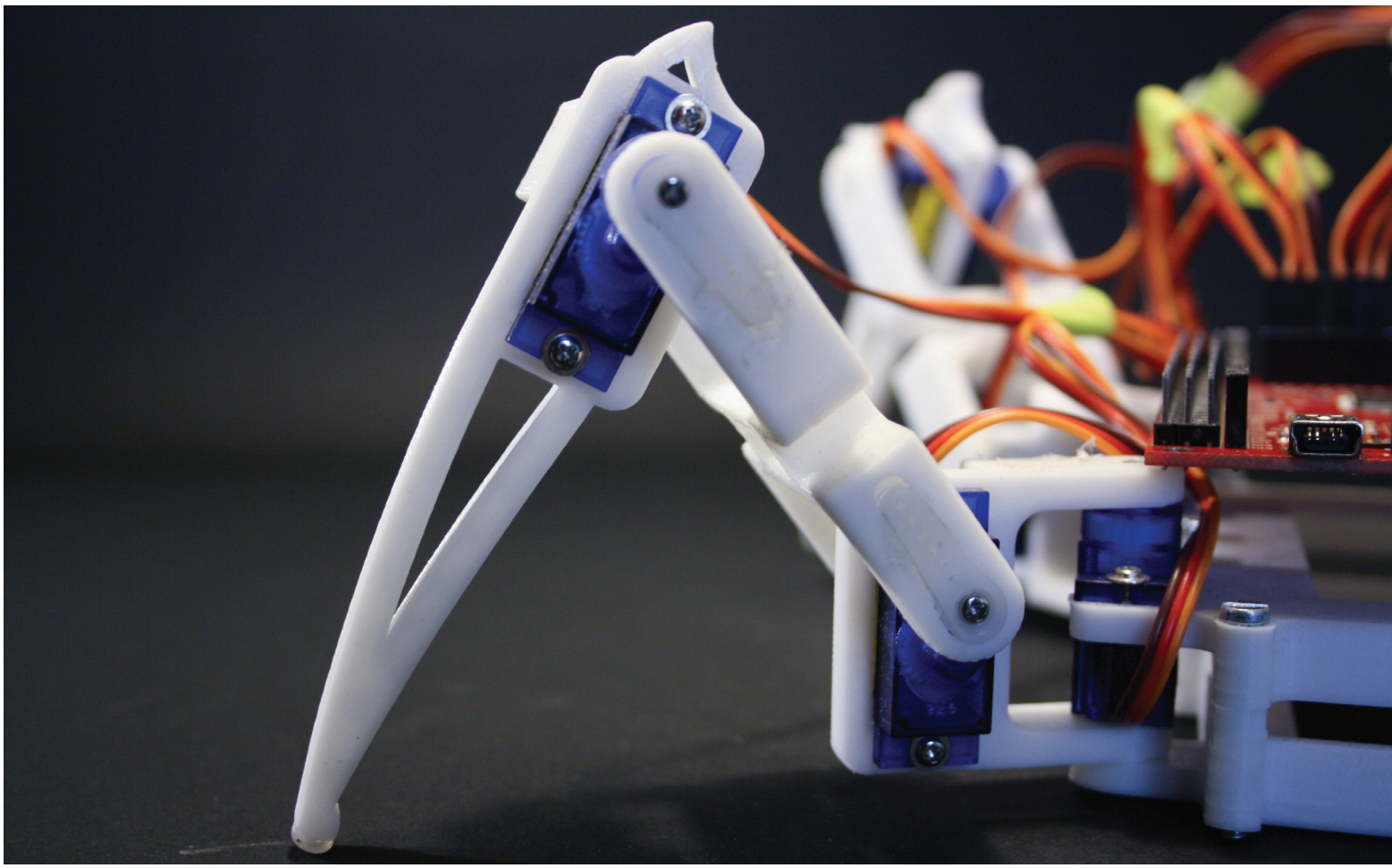


Abbildung 4.11-5 Detail

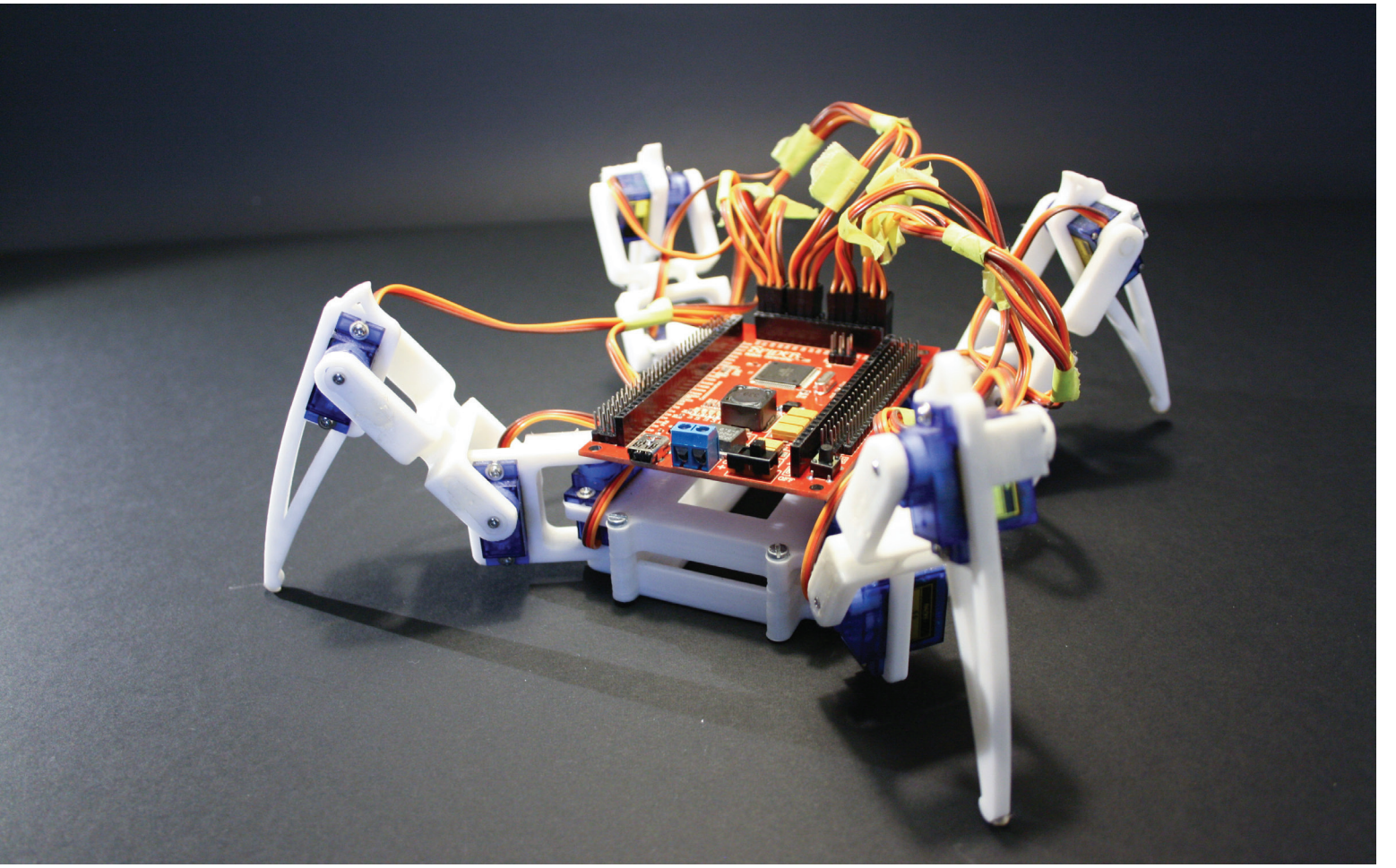


Abbildung 4.11-6 Agent mit Microcontroller

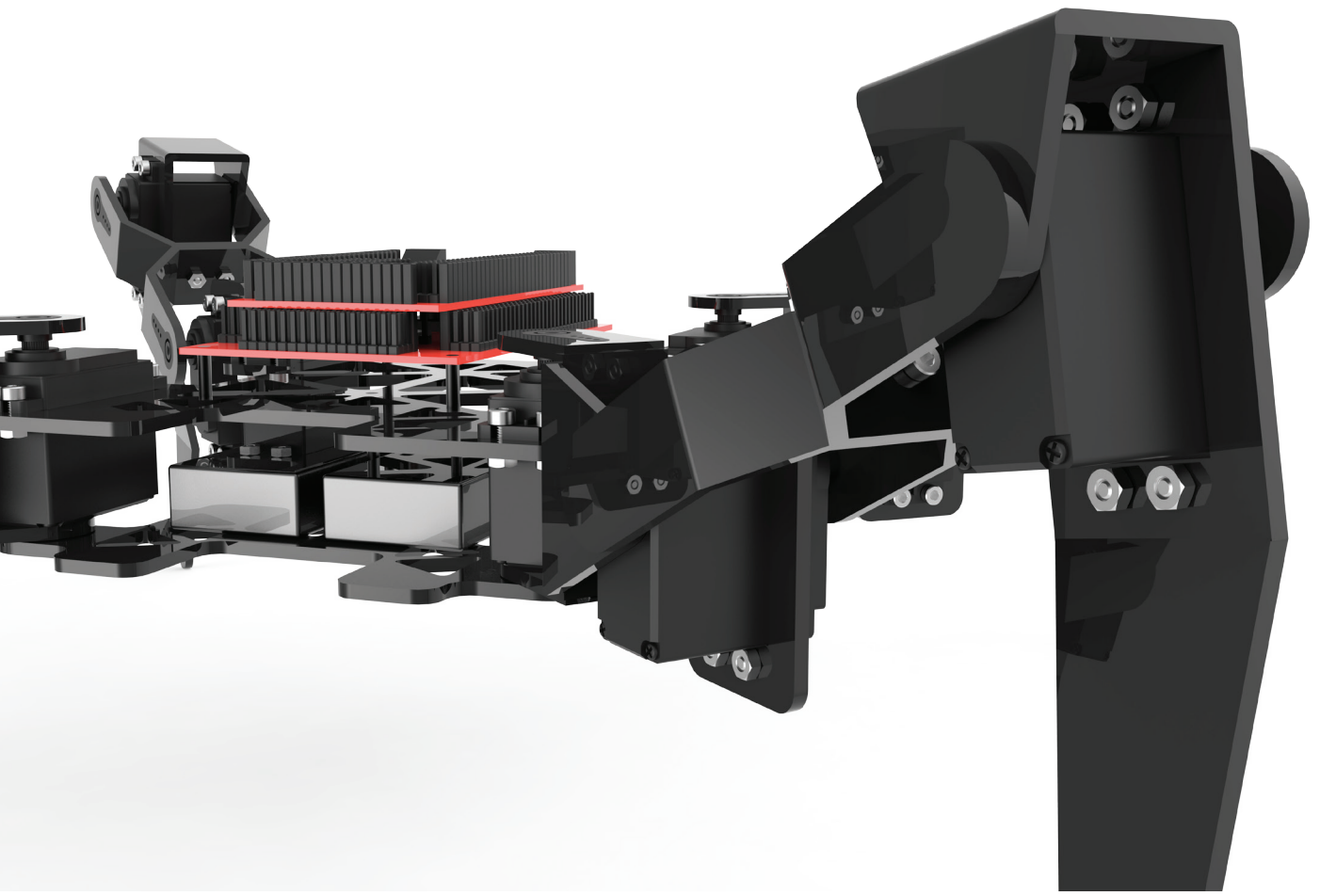


Abbildung 4.11-7 Detail

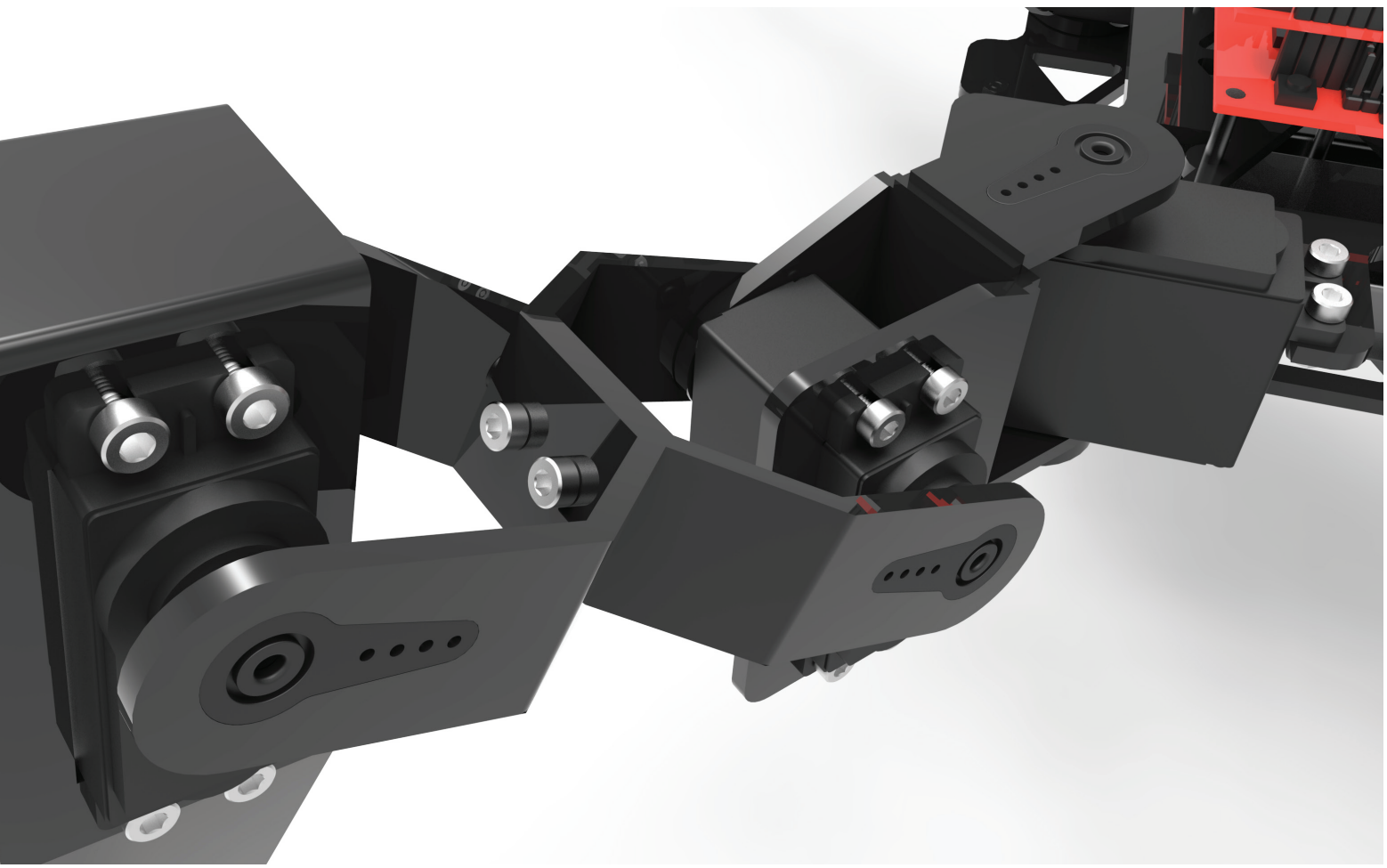


Abbildung 4.11-8 Detail Verbindung

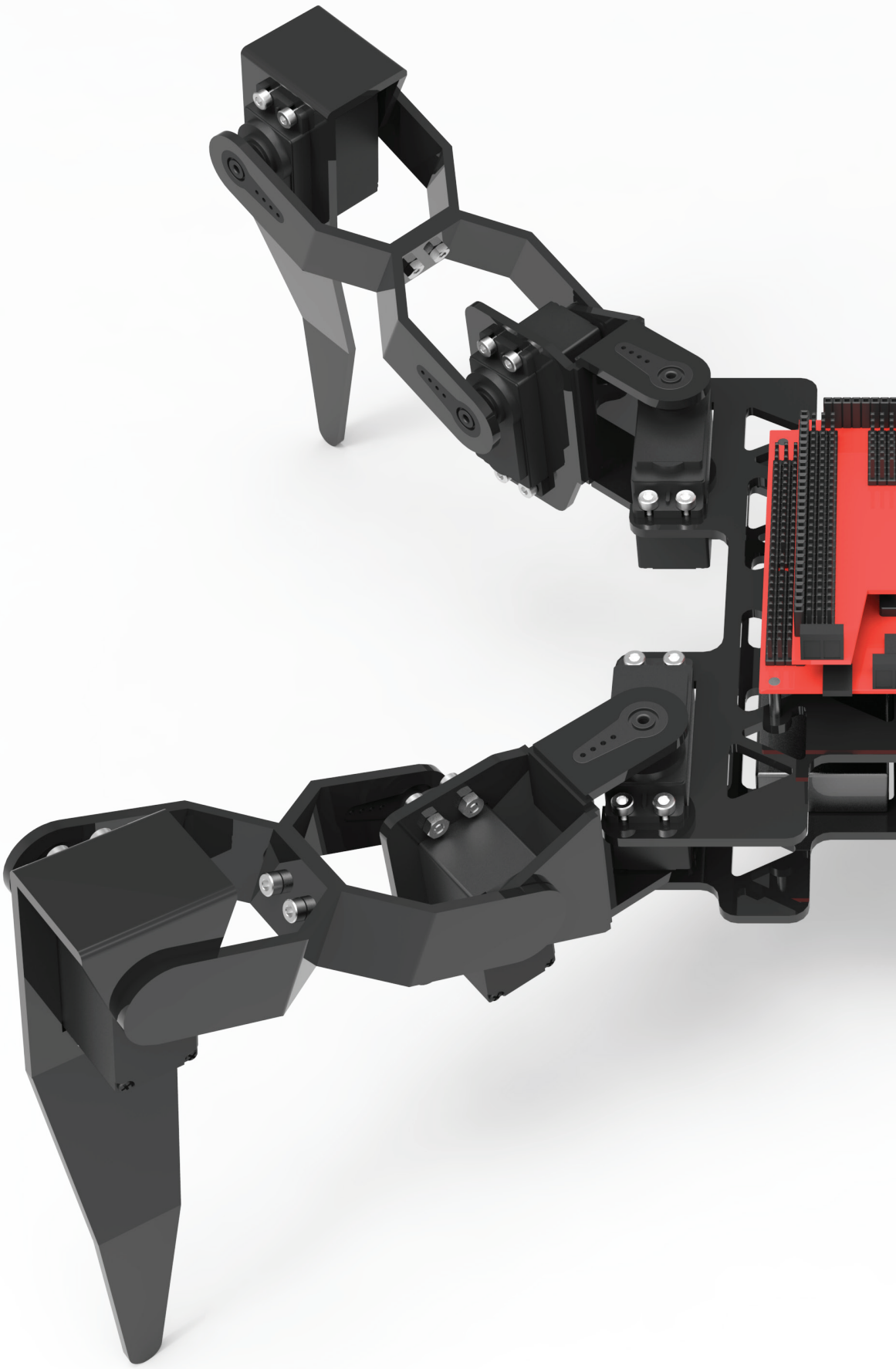


Abbildung 4.11-9 Gesamtansicht

4.12 prototyping

Bei der Roboterentwicklung wurde für das Prototyping Acrylglas-Plattenmaterial verwendet. Dieses hat den Vorteil, das es mit einem Laser zu schneiden und durch Hitze verformbar ist. Im Vergleich zum 3D-Druck ist dieses Verfahren schneller und auch günstiger.
Im ersten Schritt wurde der Prototyp in Rhinoceros 3D entwickelt. Dann wurde der 3D-Entwurf in 2D-Laserpläne abgewickelt und mithilfe eines Lasercutters ausgeschnitten. Im letzten Schritt wurden die ausgeschnittenen Teile mit Hilfe einer Heißluftpistole gezielt erwärmt und mithilfe von Schablonen in ihre dreidimensionale Form gebracht.

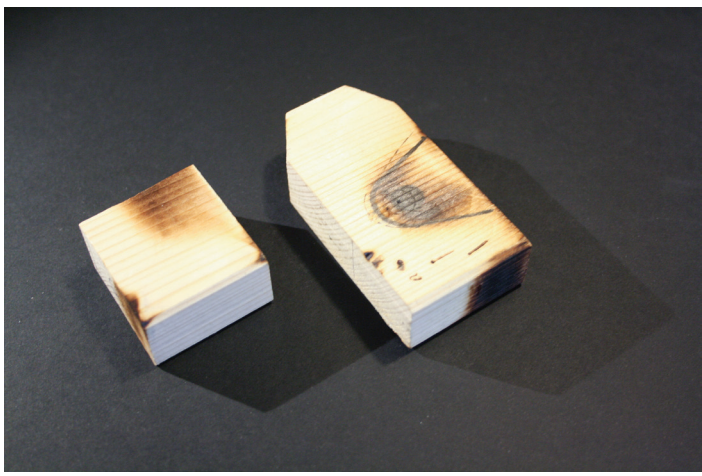
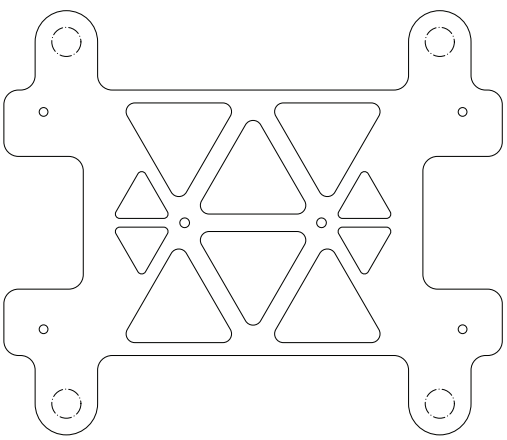
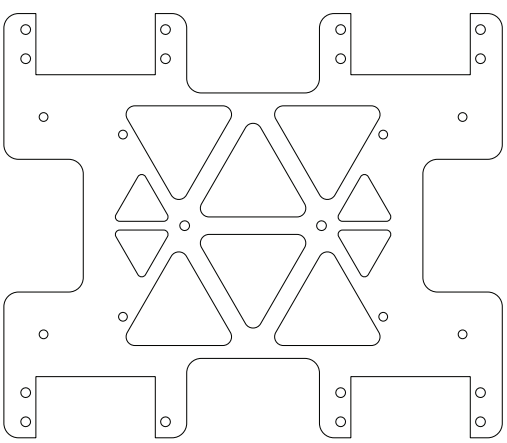


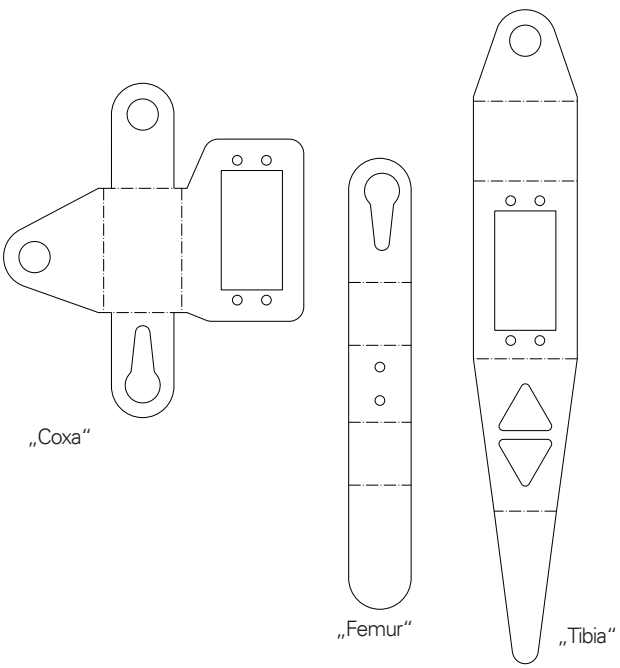
Abbildung 4.12-1 Schablonen



„Bauchplatte“



„Rückenplatte“



Zeichnung 4.12-1 Einzelteile des Roboters

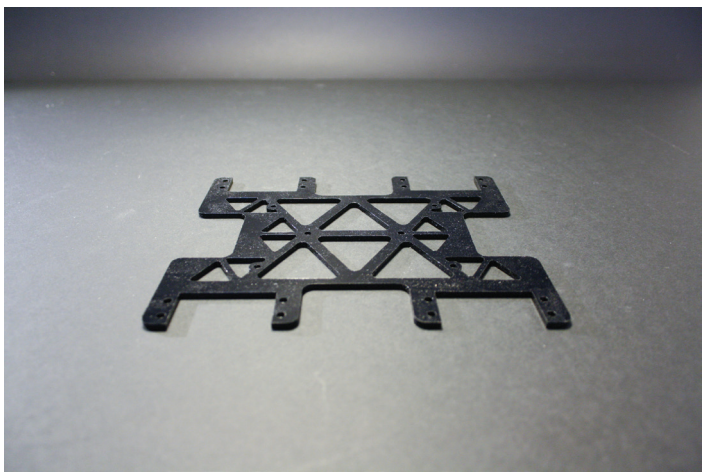


Abbildung 4.12-2 Rückenplatte

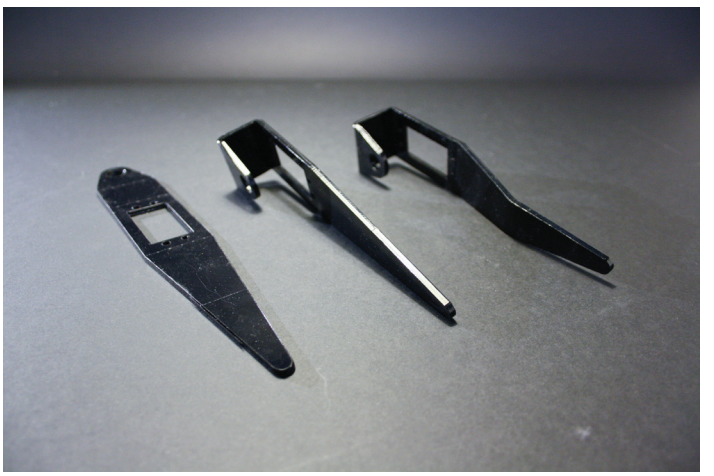


Abbildung 4.12-3 „Tibia“

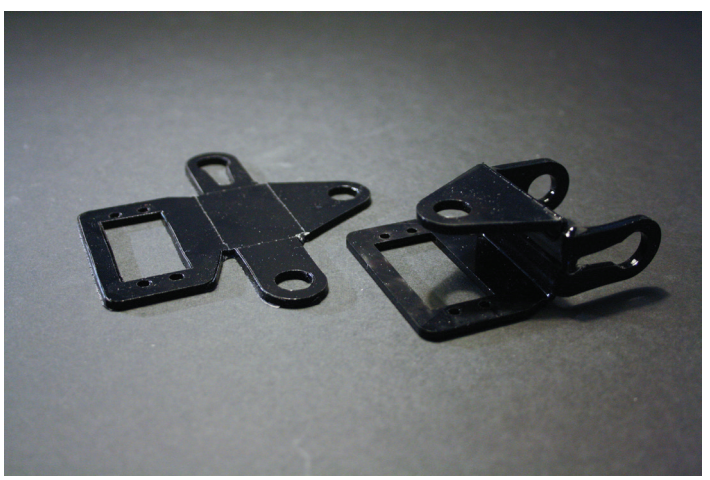


Abbildung 4.12-4 „Coxa“

Abbildung 4.12-5 „Femur“

4.13 werkzeuge

Die Einheiten können mit verschiedenen Werkzeugen ausgestattet werden. Das besondere dabei ist das jedes Bein auch gleichzeitig Werkzeug ist. Zum Beispiel:

- Druckwerkzeug
- Sammelgreifer
- Fräswerkzeug
- etc.

Der Prototyp verfügt über ein Druckwerkzeug, welches aus sich aus einem Controller, einem Vorschubmotor und einem Druckkopf besteht.

Abbildung 4.13-1 Detail Druckkopf

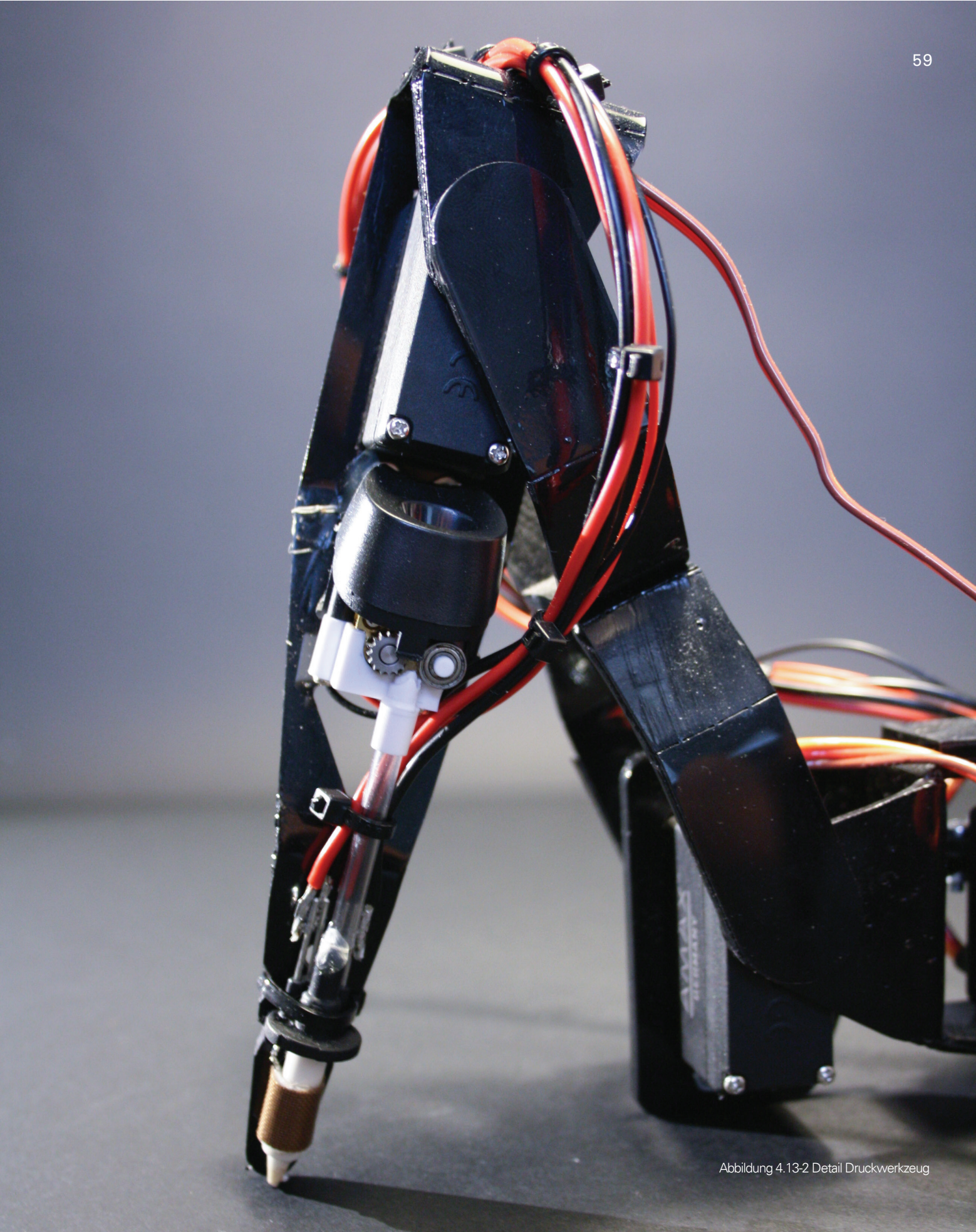


Abbildung 4.13-2 Detail Druckwerkzeug

4.14 sensoren

Die Einheiten können mit verschiedenen Sensoren zur Erkundung ihrer Umgebung ausgestattetwerden Zum Beispiel:

- GPS
- Abstandsmesser
- Kameras zum scannen von Sammelgut

Die Agenten sind mit Hilfe von Sensoren in der Lage ihre Umgebung wahrzunehmen und auf sie zu reagieren. Die einegsetzen Sharp GP2Y0A02Y Abstandsmessssensoren können dies sehr präzise bewerkstelligen. Sie kommen dann zum Einsatz wenn sich der Agent im „Exploremode“ befindet. Das Skript auf der nächsten Seite stellt die Abläufe im „Exploremode“ da.

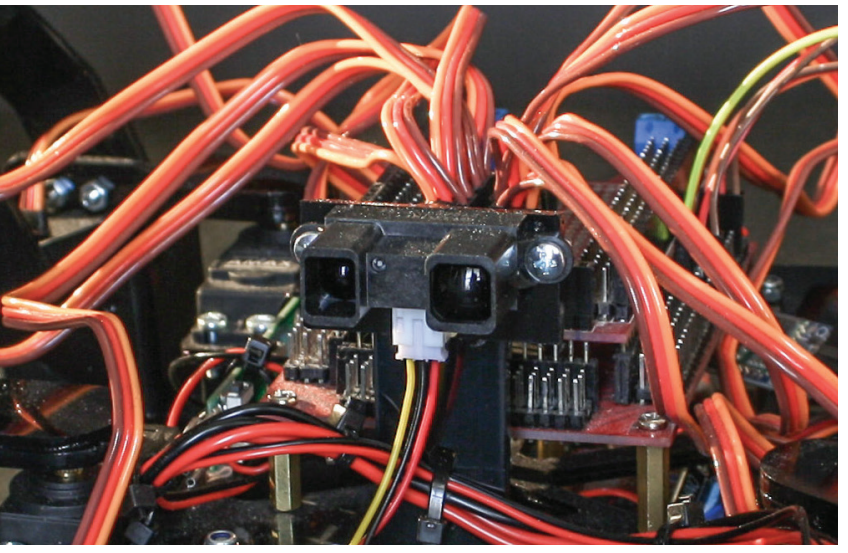


Abbildung 4.14-1 Detail Obstacle Avoidance Sensor

```
#include <SharpIR.h>

#define ir A10
#define model 20150
// ir: the pin where your sensor is attached
// model: an int that determines your sensor:  1080 for GP2Y0A21Y
//                                            20150 for GP2Y0A02Y
//                                            (working distance range according
to the datasheets)

SharpIR SharpIR(ir, model);
long randNumber;

void explore(void) {

  int dis = SharpIR.distance();

  if (dis < 30) {
    step_forward(3);
    delay(1500);

    randNumber = random(1, 3);

    if (randNumber == 1) {
      turn_right(5);
      delay(2000);
    }

    else if (randNumber == 2) {
      turn_left(5);
      delay(2000);
    }
  }

  else {
    step_back(1);
  }
}
```

Skript 4.14-1 Detail Programmierung Obstacle Avoidance Sensor, Sprache: C++

4.15 kunstoffarten

Kunststoffe zählen zu den wichtigsten Produkten weltweit. Mittlerweile existiert eine Vielzahl an Kunstoffen für verschiedenste Anwendungen. In der Produktion dominieren sechs Stoffe mit einem Anteil von 90% den Markt und damit auch unsere Meere und Strände.
Alle Stoffe werden nach einem weltweit einheitlichen System klassifiziert und mittels Symbolen kenntlich gemacht. Die verschiedenen Kunststoffklassen weisen unterschiedliche Merkmale und Eigenschaften im Bezug auf ihre Wiederverwertbarkeit und Zersetzungszeit auf. So benötigen bestimmte Kunststoffe bis zu 600 Jahre bis sie vollständig zersetzt sind.
Für die Wiederverwendung als Druckmaterial kommen vor allem PET, PE und ABS sowie Mischungen aus selbigen in Betracht.

Konkret bedeutet das für das vorliegende Szenario, dass durch die einheitliche Kennzeichnung der Stoffklassen eine Trennung in wiederverwertbare und nicht-wiederverwertbare Materialien vereinfacht wird und zum anderen von einer nicht unerheblichen Menge an potentiellem Druckmaterial ausgegangen werden kann da sowohl PET als auch PE zu den dominierenden sechs Stoffprodukten gehören.
Im Falle des exemplarischen Roboter-Prototypen, kommt der Kunststoff PLA als Druckmaterial zum Einsatz.

Reprap, 2014

PET		polyethylene terephthalate	clear, tough, solvent resistant, barrier to gas and moisture, softens at 80°	Soft drink, water bottles, salad domes, bisquit trays, food containers	yellow flame little smoke
HDPE		high-density polyethylene	Hard to semi-flexible, resistant to chemicals and moisture, waxy surface, softens at 75°	Shopping bags, freezer bags, milk bottles, juice bottles, iceacream containers, shampoo, crates	difficult to ignite smells like candle
PVC		polyvinyl chloride	Strong, tough, can be clear and solvent, softens at 60°	Cosmetic containers, electrical condult, plumbing pipes, blister packs, roof sheeting, garden hose	yellow flame green spurts
LDPE		low-density polyethylene	Soft, flexible, waxy surface, scratches easily, softens at 70°	Cling wrap, garbage bags, squeeze bottles, refuse bags, mulch film	difficult to ignite smells like candle
PP		polypropylene	Hard but still flexible, waxy surface, translu-cent, withstands solvents, softens at 140°	Bottles, icecream tubes, straws, flower-pots, dishes, garden furniture, food containers	blue yellow tipped flame
PS		polystyrene	Clear, glassy, opaque, semi tough, softens at 95°	CD cases, plastic cutlery, imitation glass, foamed meat trays, brittle toys,	dense smoke
OTHER		all other plastics	Properties depend on the type of plastic	automotive, electron-ics, packaging	all other plastics

Tabelle 4.15-1 Kunstoffarten und Eigenschaften

4.16 druckexperimente

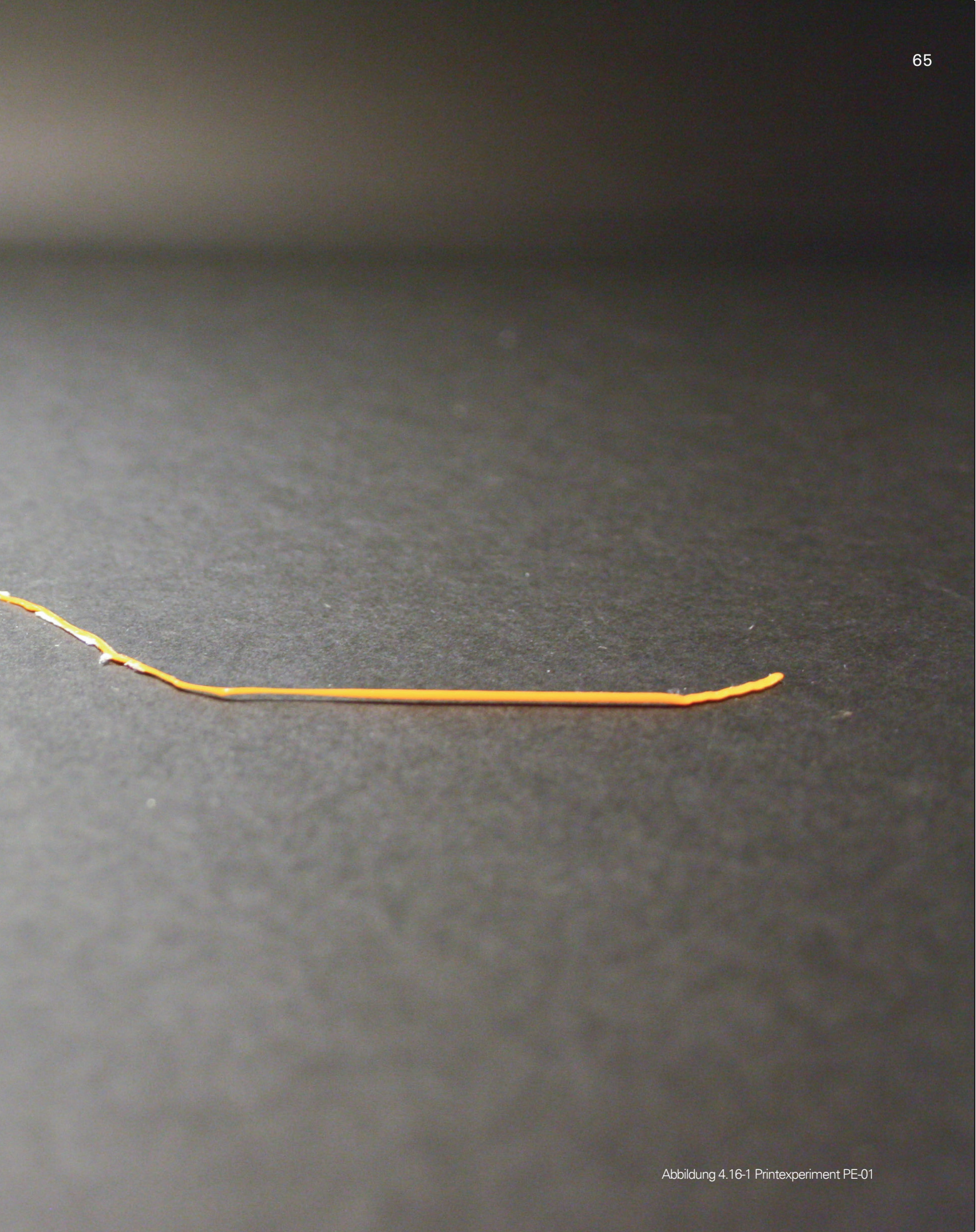
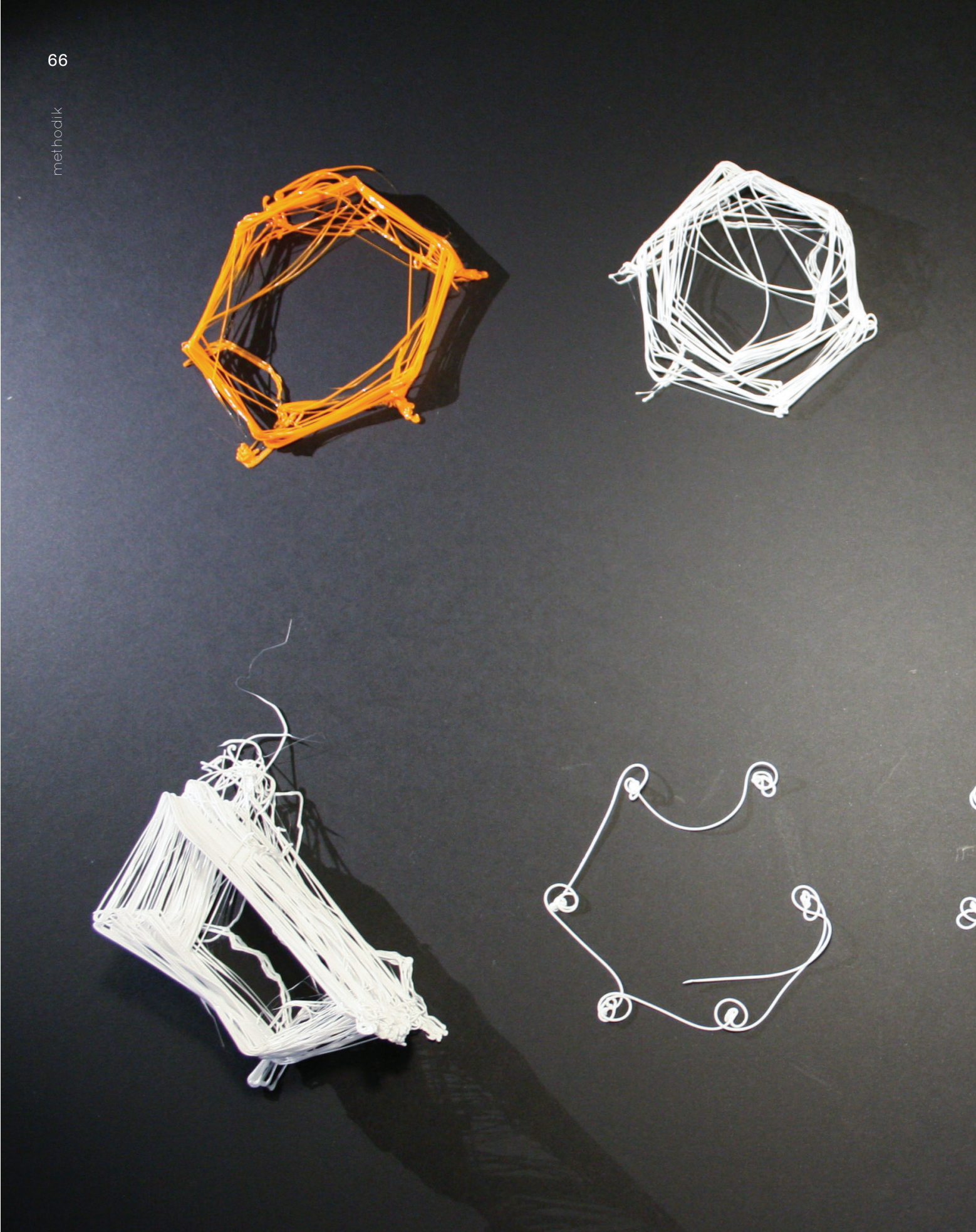


Abbildung 4.16-1 Printexperiment PE-01



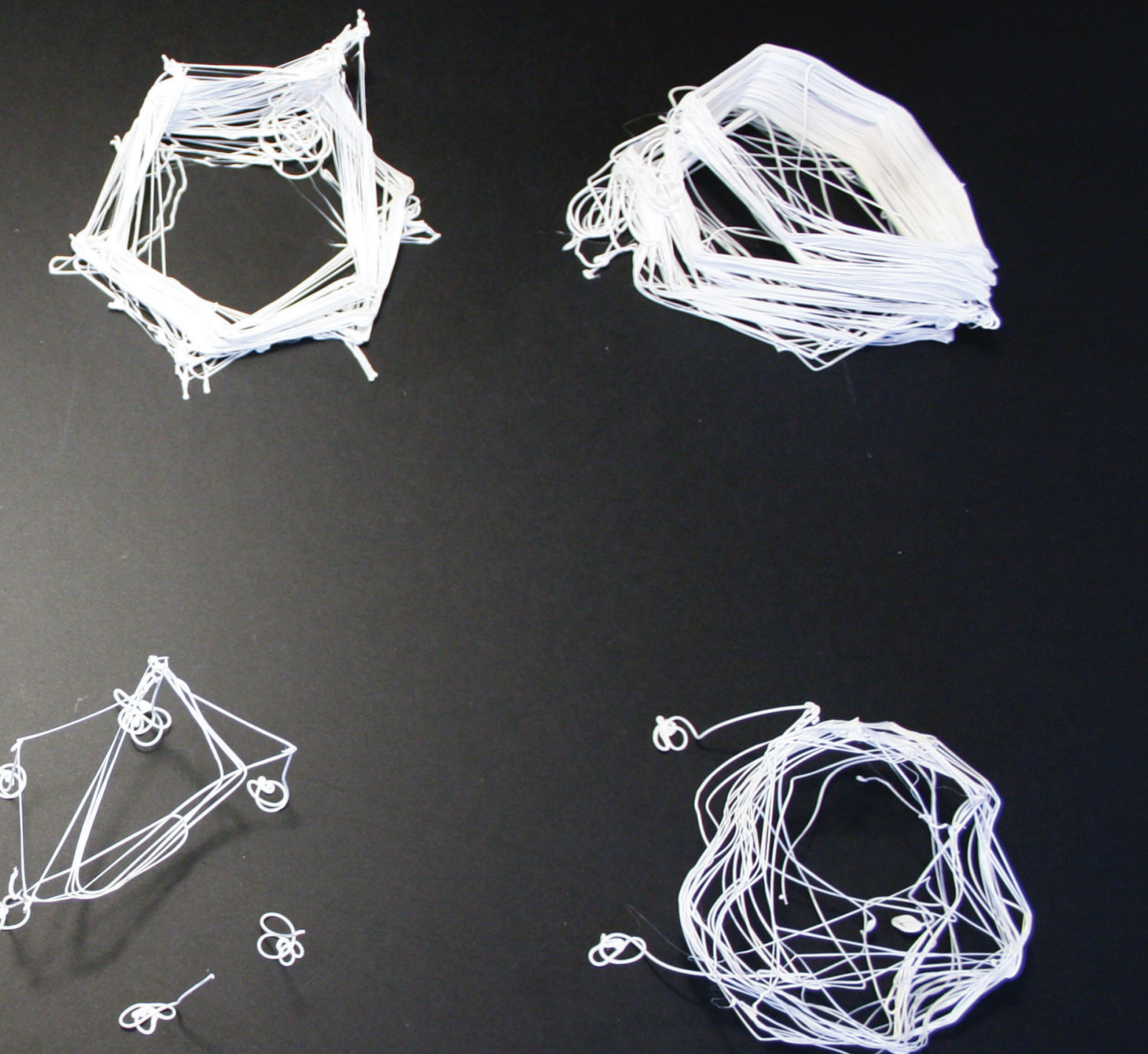


Abbildung 4.16-2 Printexperimente Übersicht

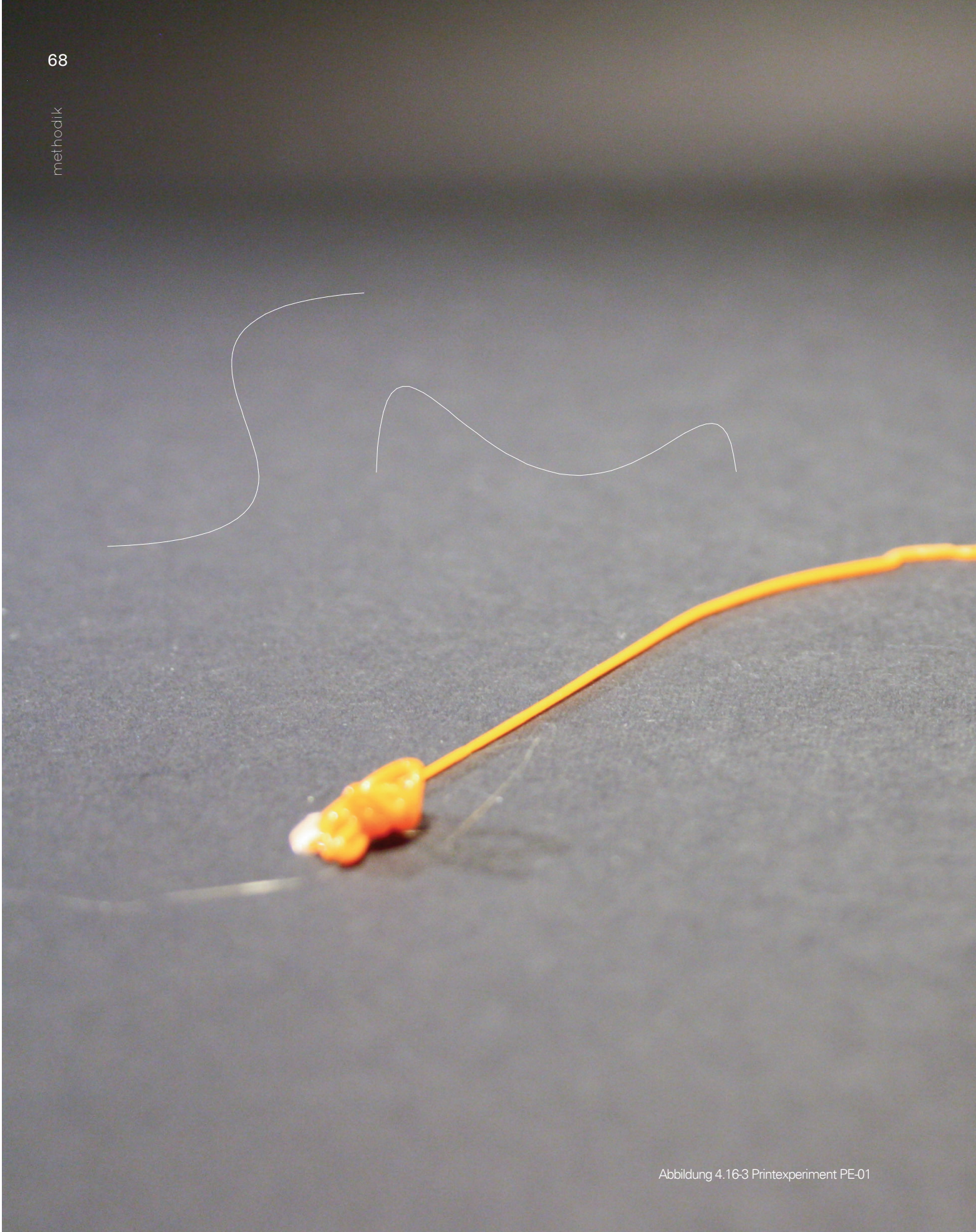
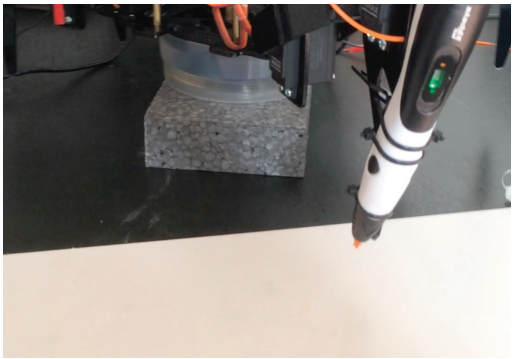
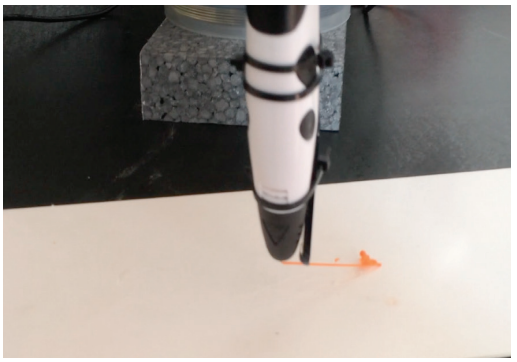
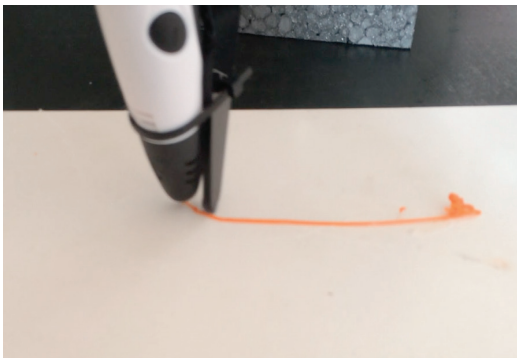
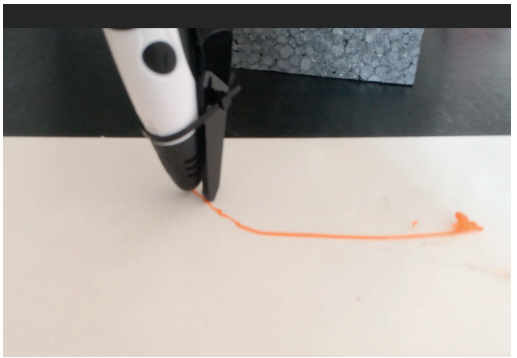
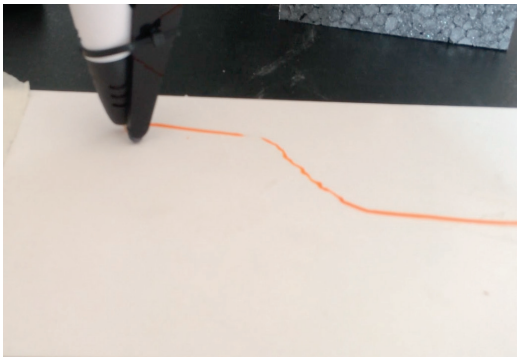
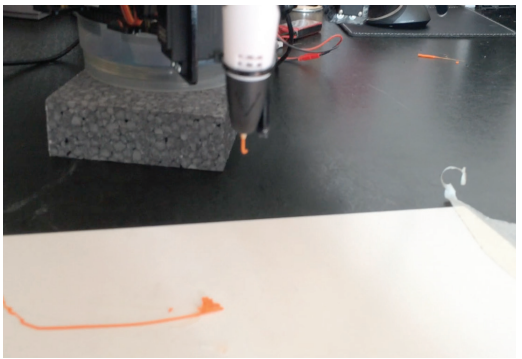


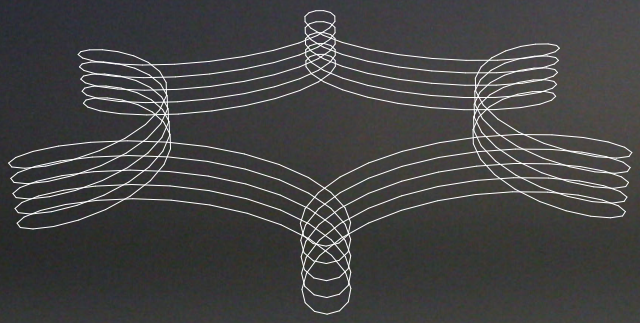
Abbildung 4.16-3 Printexperiment PE-01



Objekt ID:	PE-01
Material:	PLA 0.75mm
Druckzeit:	0:05 min
Materialmenge:	0.5g

Abbildung 4.16-4 Printexperiment PE-01 Verlauf



```
selectpath()
time.sleep(1)
printer()
time.sleep(0.5)
path()
path()
movepath_z()
path()
movepath_z()
path()
path()
movepath_z()
path()
path()
movepath_z()
path()
printer()
```

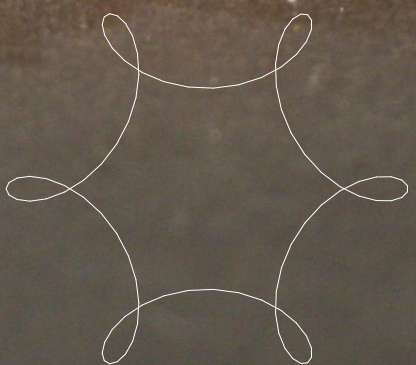
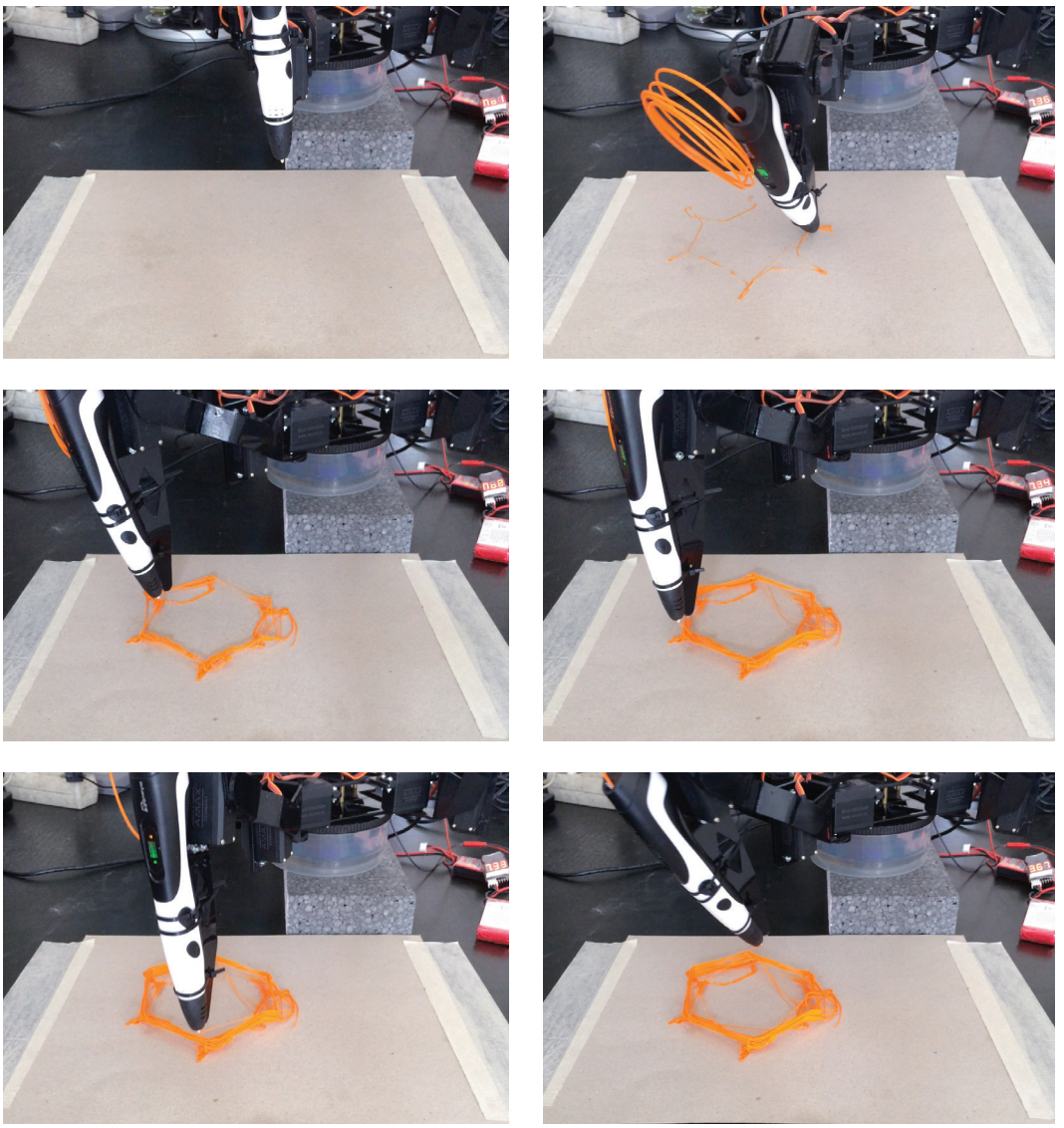


Abbildung 4.16-5 Printexperiment PE-02



Objekt ID:	PE-02
Material:	PLA 0.75mm
Druckzeit:	6:32 min
Materialmenge:	2g

Abbildung 4.16-6 Printexperiment PE-02 Verlauf

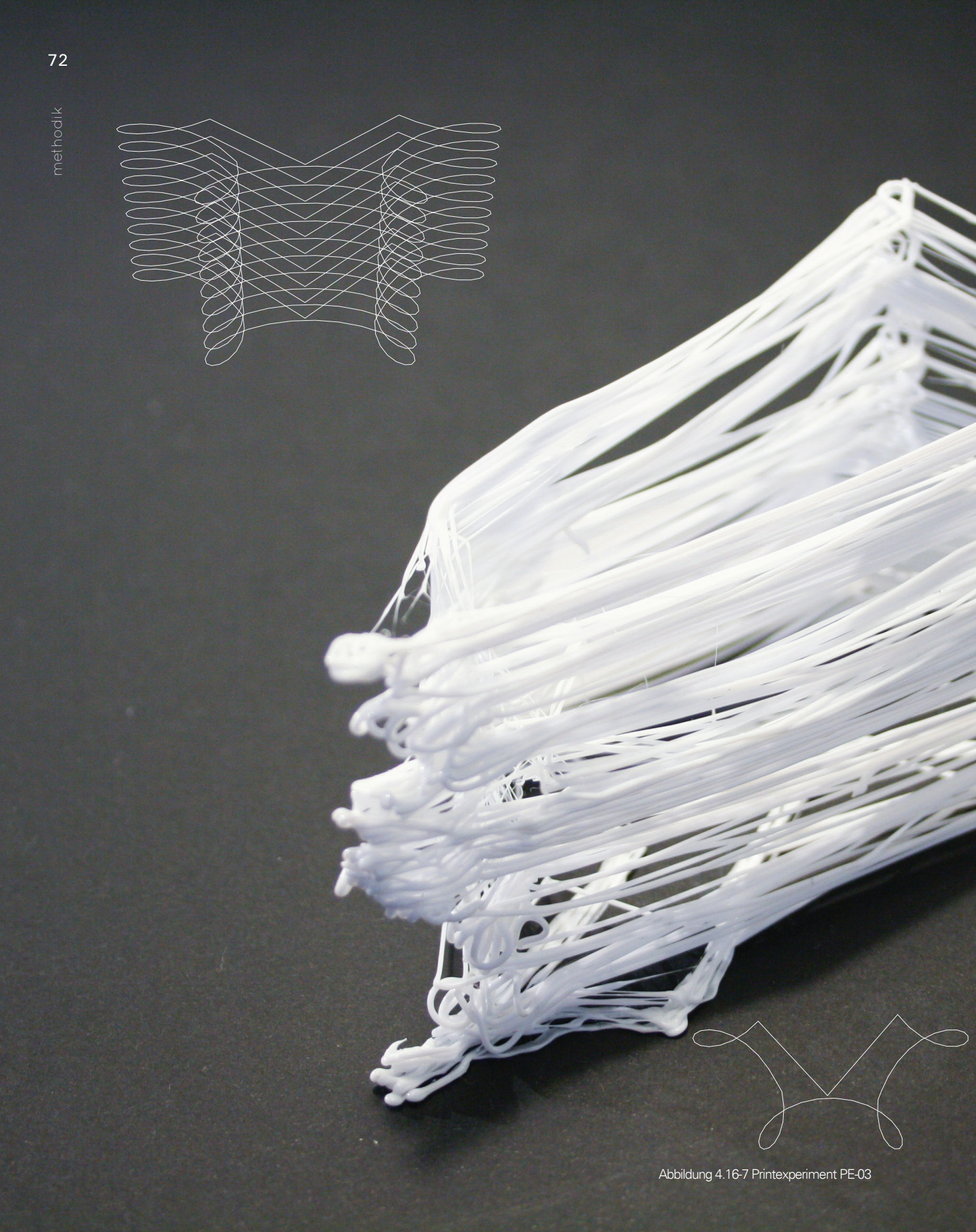
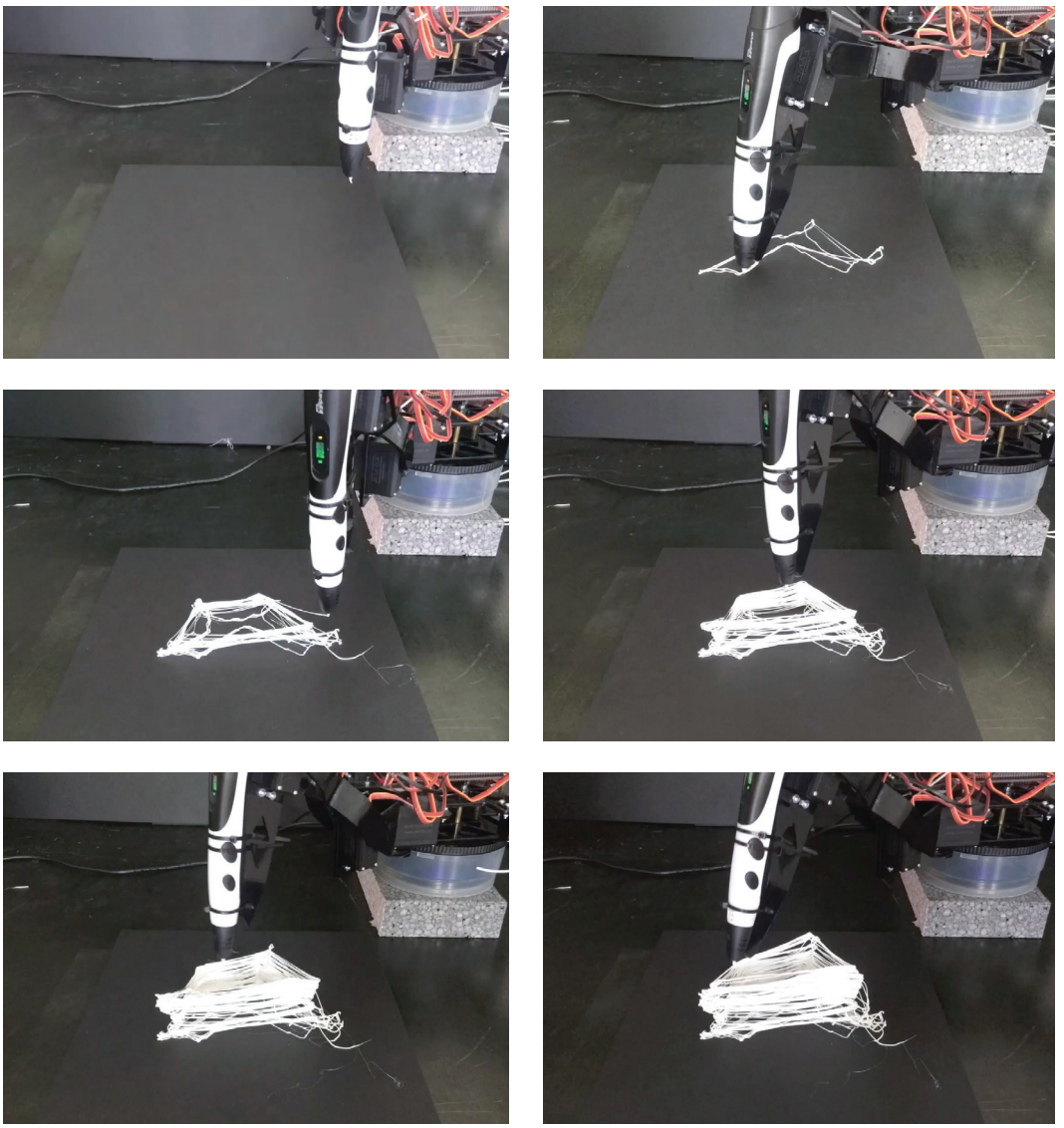


Abbildung 4.16-7 Printexperiment PE-03



Objekt ID:	PE-03
Material:	PLA 0.75mm
Druckzeit:	13:17 min
Materialmenge:	7g

Abbildung 4.16-8 Printexperiment PE-03 Verlauf

Objekt ID:	PE-04
Material:	PLA 0.75mm
Druckzeit:	15:21 min
Materialmenge:	9g

Abbildung 4.16-9 Printexperiment PE-04 Detail

Objekt ID:	PE-05
Material:	PLA 0.75mm
Druckzeit:	10:48 min
Materialmenge:	5g

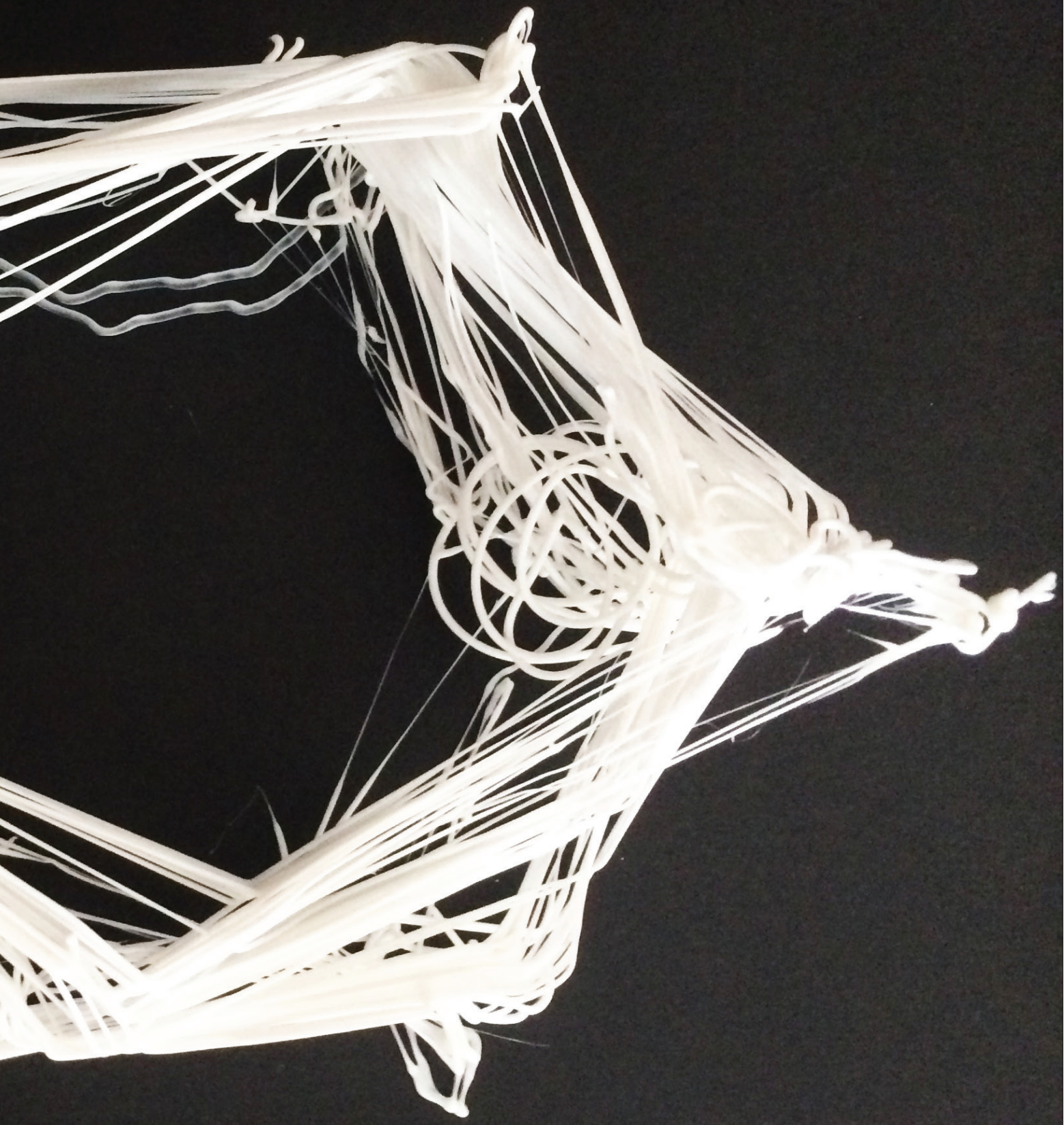
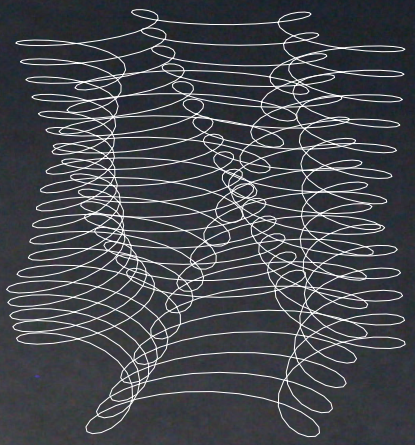


Abbildung 4.16-10 Printexperiment PE-03



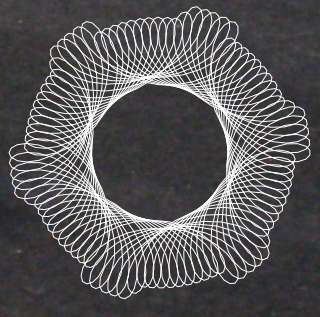
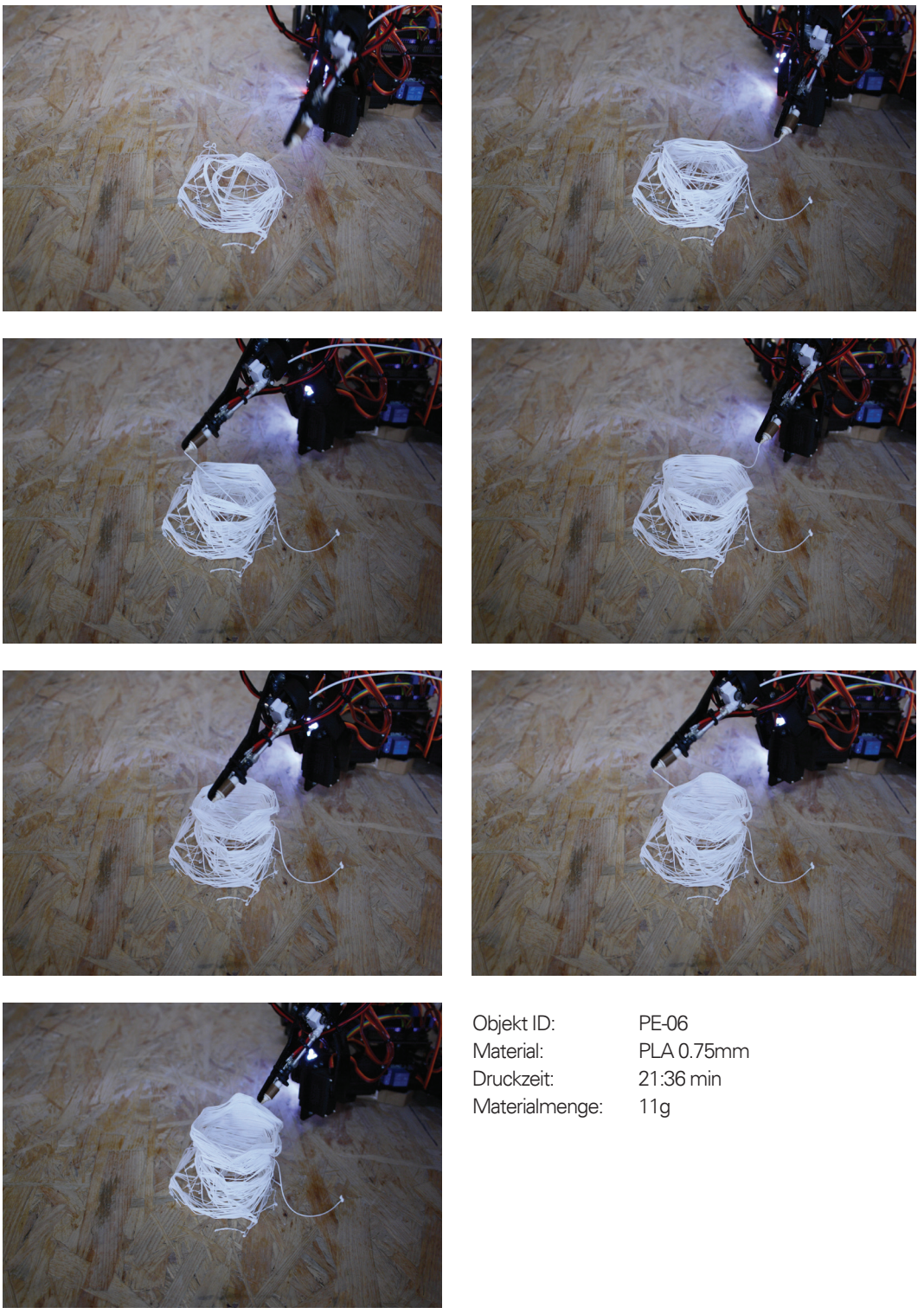


Abbildung 4.16-11 Printexperiment PE-04



Objekt ID:	PE-06
Material:	PLA 0.75mm
Druckzeit:	21:36 min
Materialmenge:	11g

Abbildung 4.16-12 Printexperiment PE-04 Verlauf

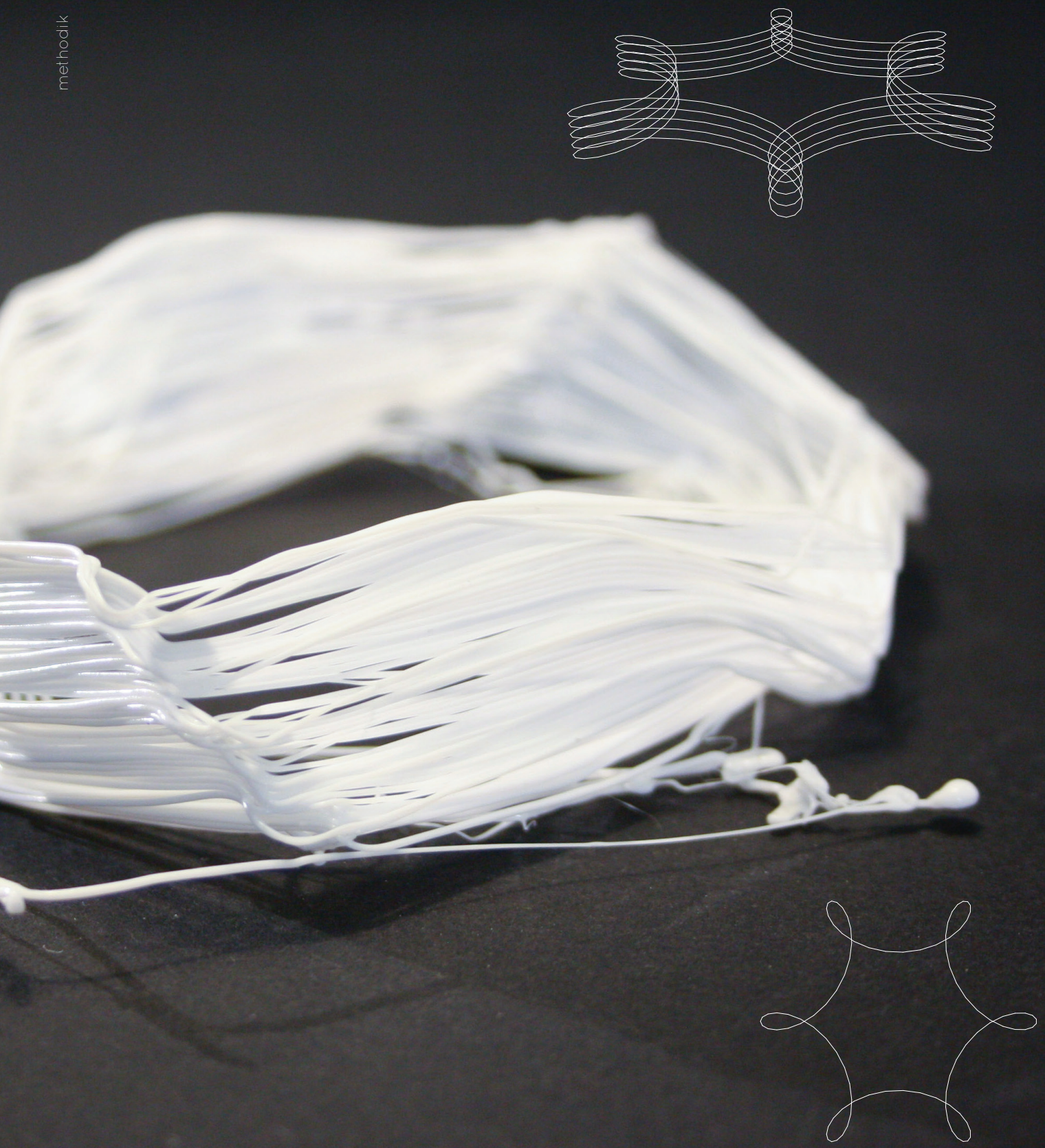
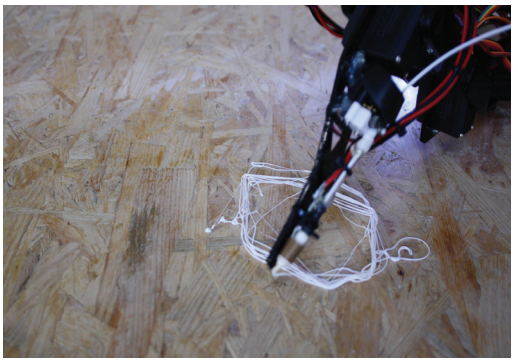
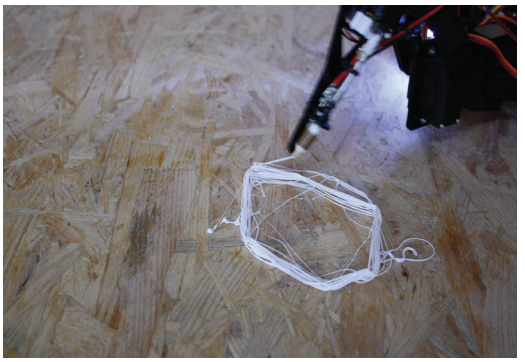
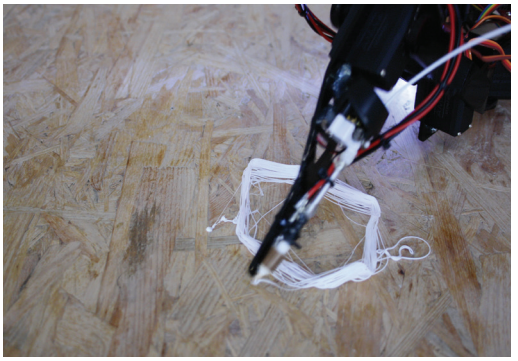
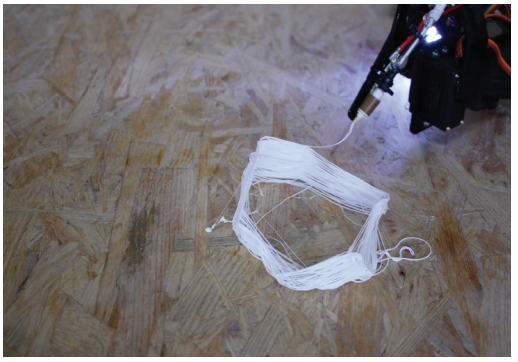


Abbildung 4.16-13 Printexperiment PE-04



Objekt ID:	PE-07
Material:	PLA 0.75mm
Druckzeit:	11:35 min
Materialmenge:	5g

Abbildung 4.16-14 Printexperiment PE-04 Verlauf

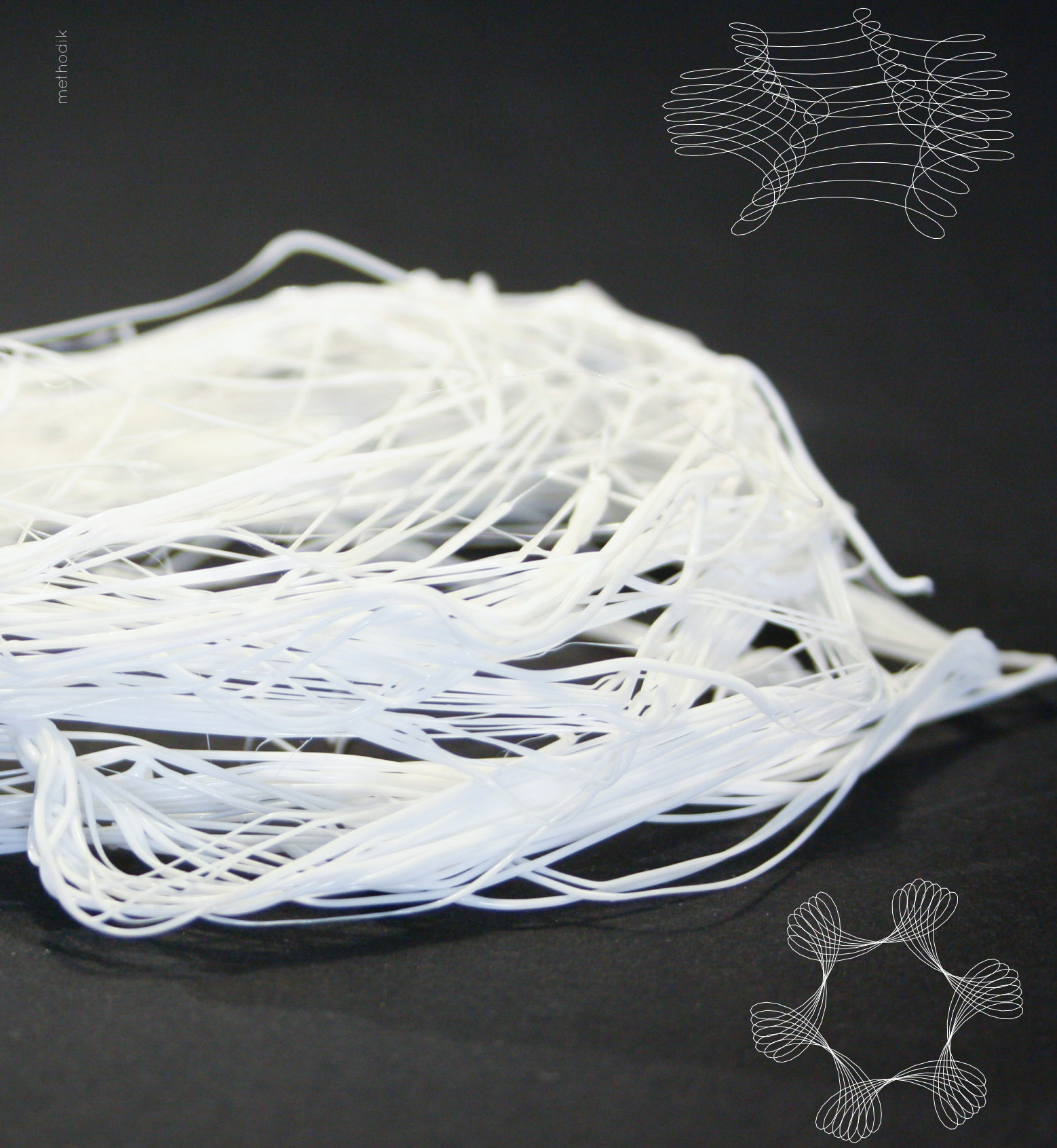
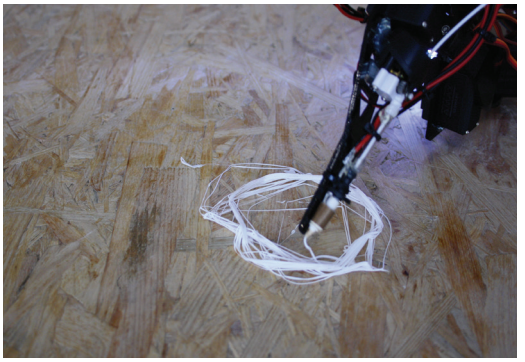
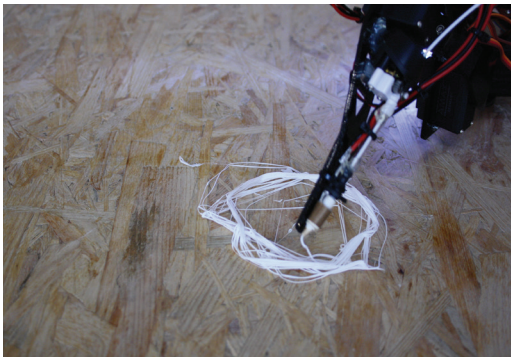
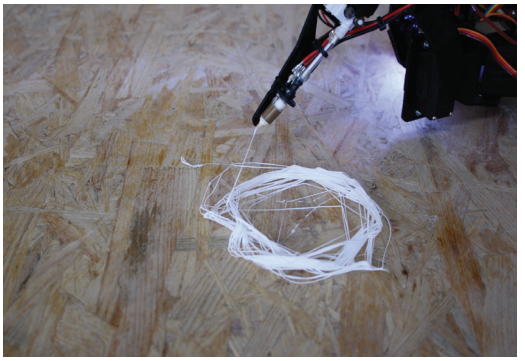
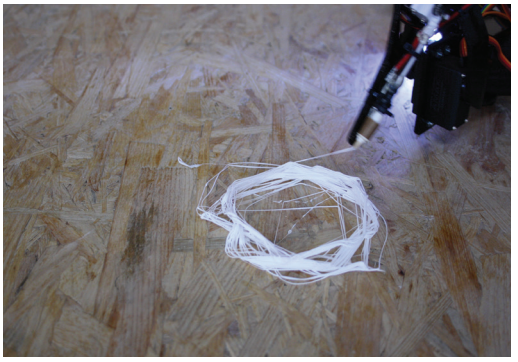
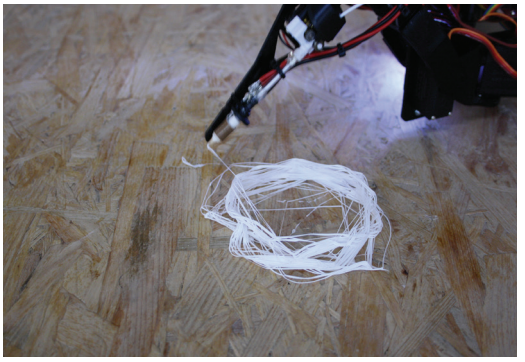
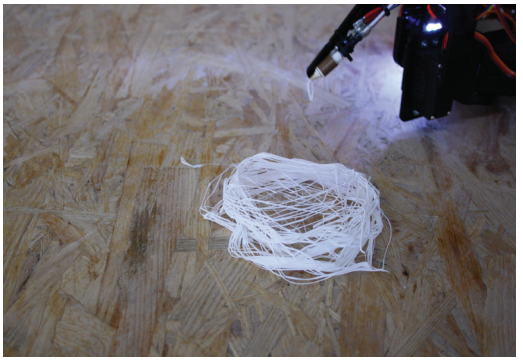


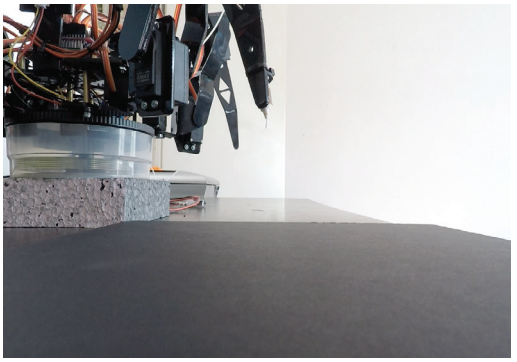
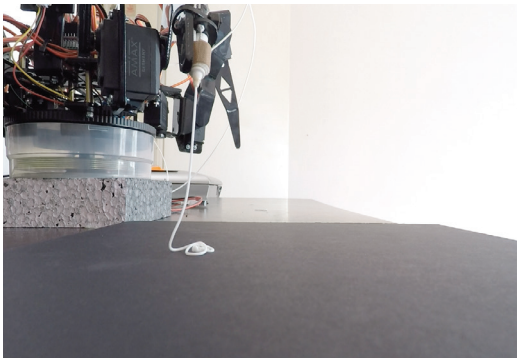
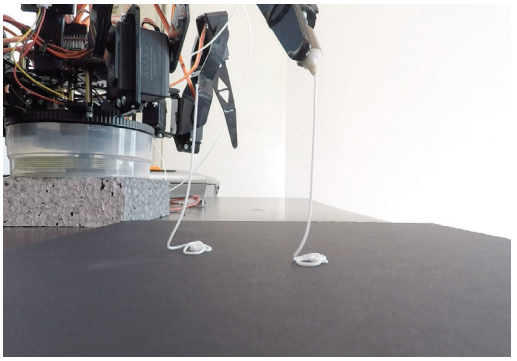
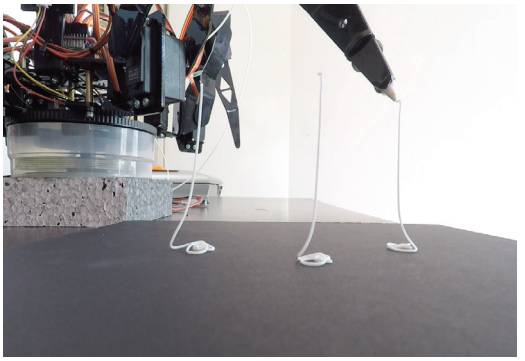
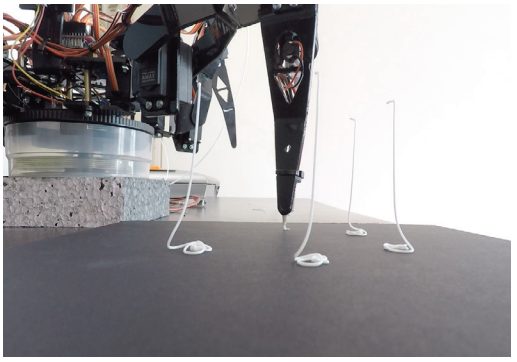
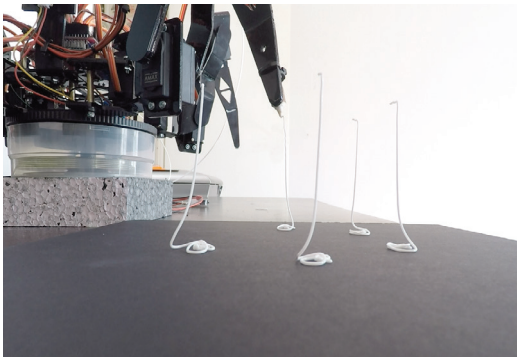
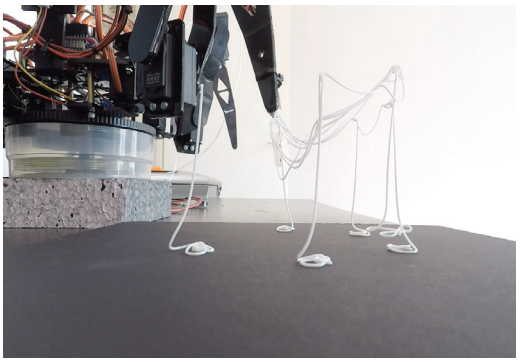
Abbildung 4.16-15 Printexperiment PE-04

Objekt ID:	PE-08
Material:	PLA 0.75mm
Druckzeit:	11:24 min
Materialmenge:	5g

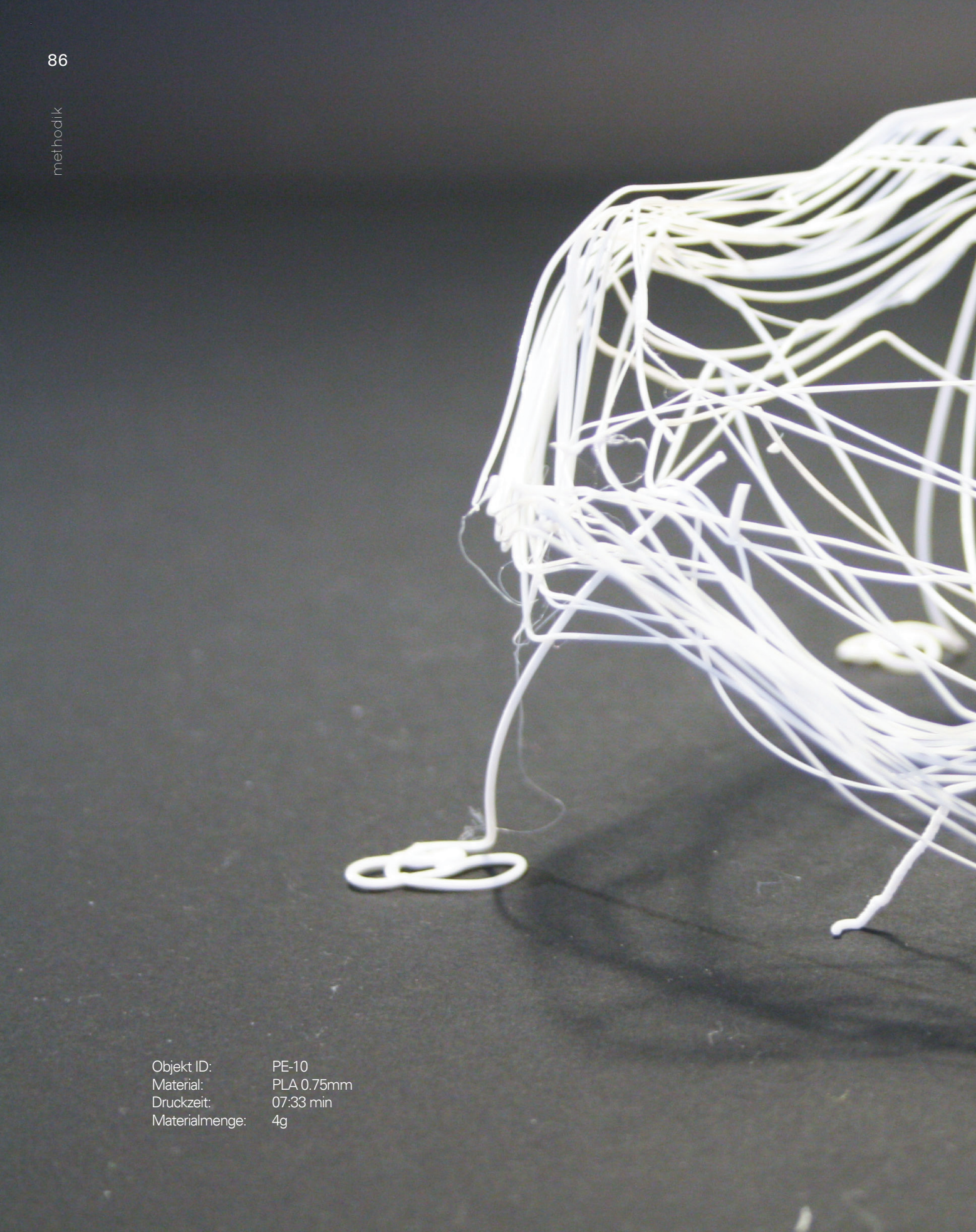
Abbildung 4.16-16 Printexperiment PE-04 Verlauf

Abbildung 4.16-17 Printexperiment PE-09

Objekt ID:	PE-09
Material:	PLA 0.75mm
Druckzeit:	03:47 min
Materialmenge:	2g

Abbildung 4.16-18 Printexperiment PE-04 Verlauf



Objekt ID:	PE-10
Material:	PLA 0.75mm
Druckzeit:	07:33 min
Materialmenge:	4g

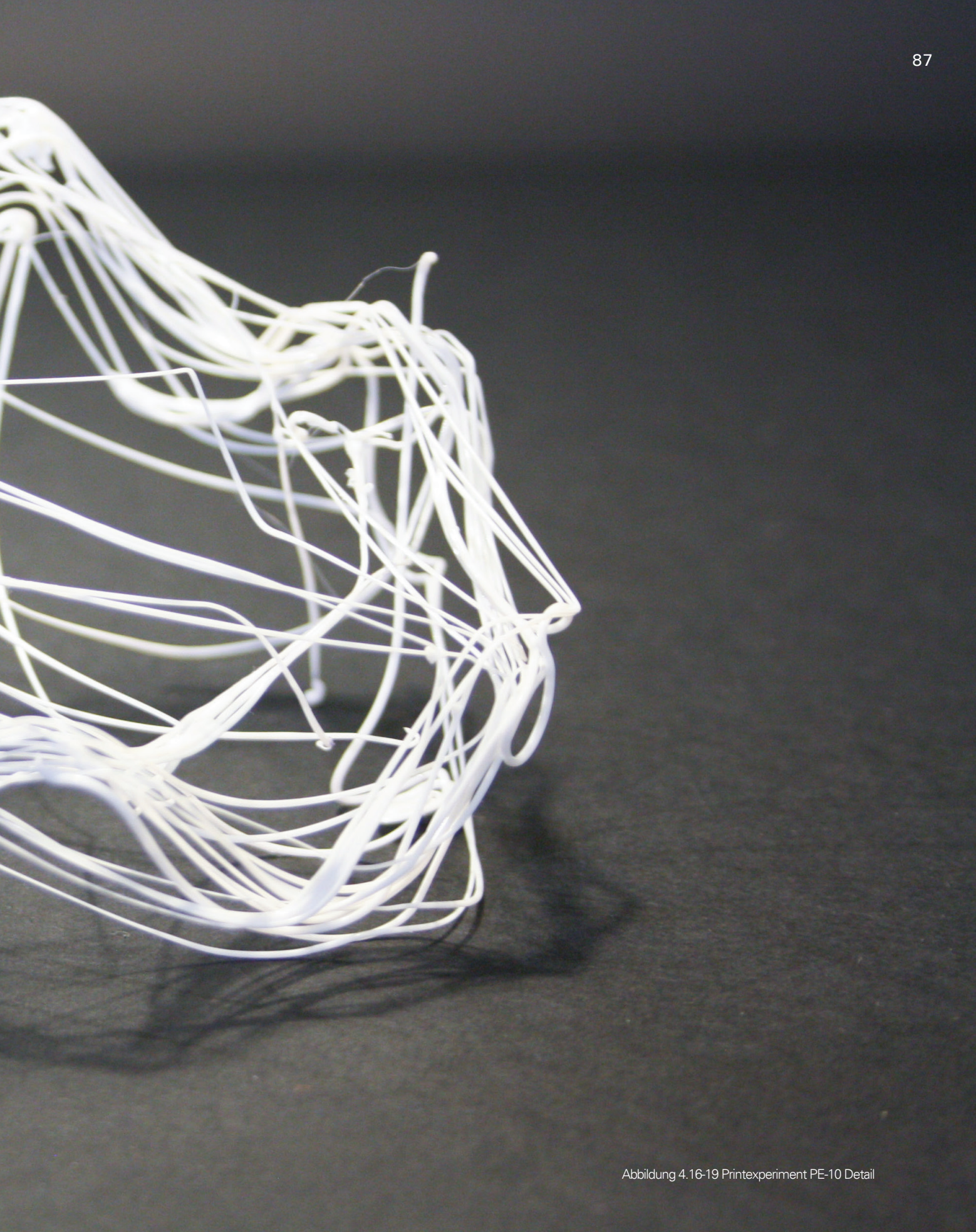
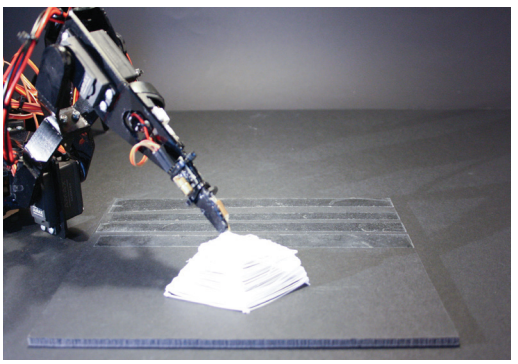


Abbildung 4.16-19 Printexperiment PE-10 Detail

Abbildung 4.16-20 Printexperiment PE-10 Detail

Abbildung 4.16-21 Printexperiment PE-11



Objekt ID:	PE-11
Material:	PLA 0.75mm
Druckzeit:	08:32 min
Materialmenge:	5g

Abbildung 4.16-22 Printexperiment PE-11 Verlauf

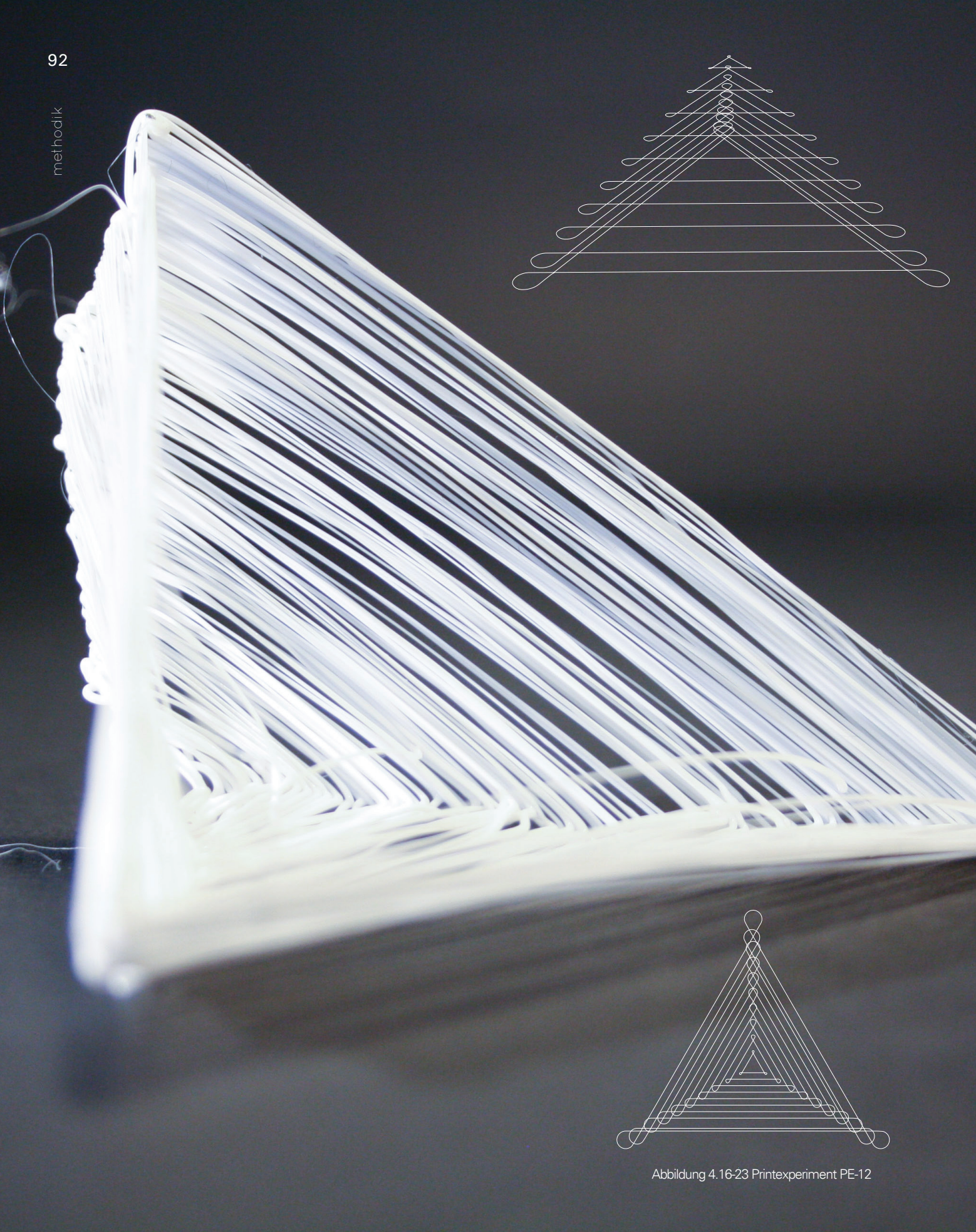
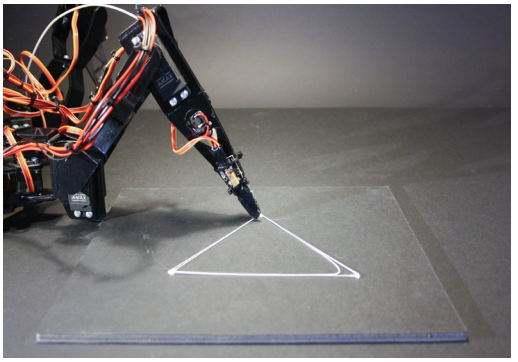
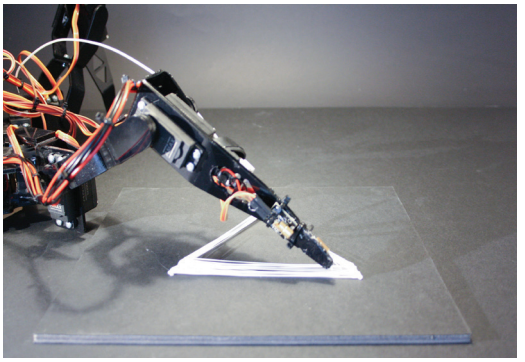
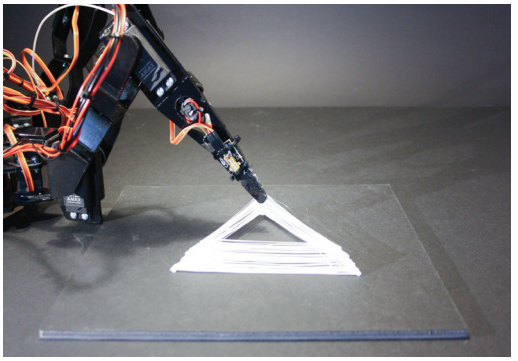
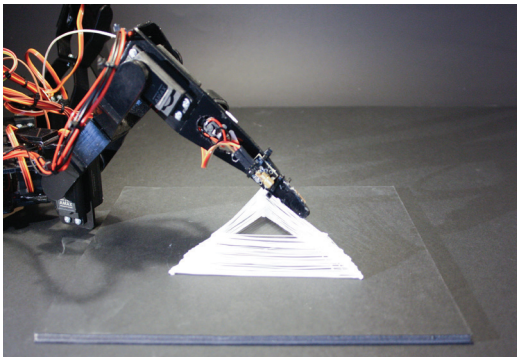
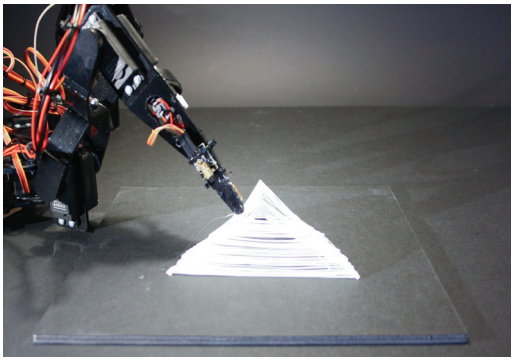
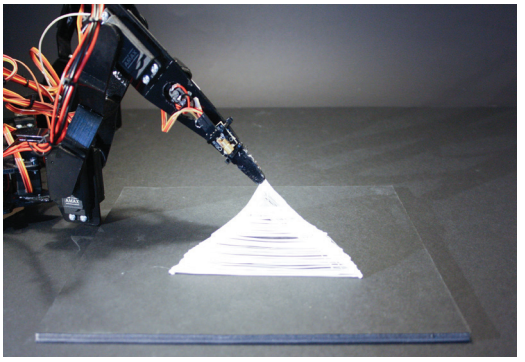
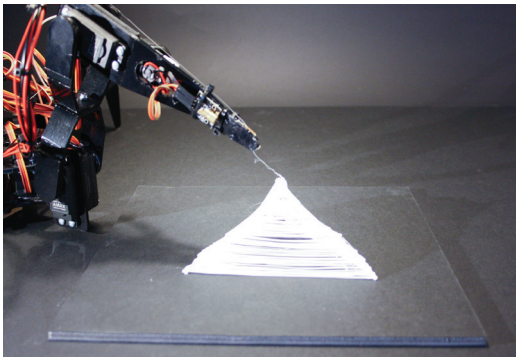


Abbildung 4.16-23 Printexperiment PE-12

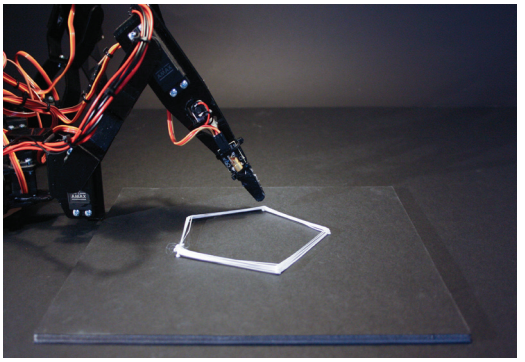
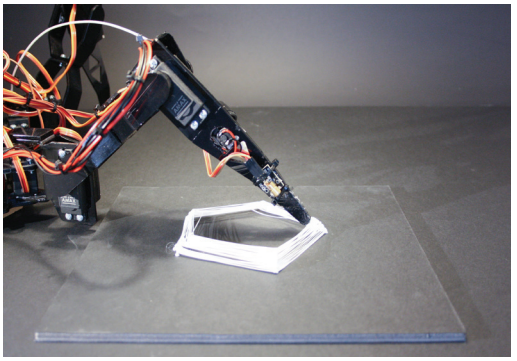
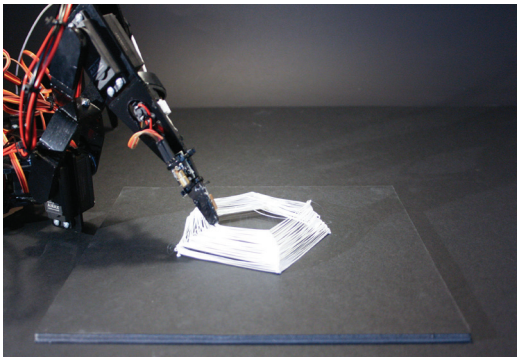








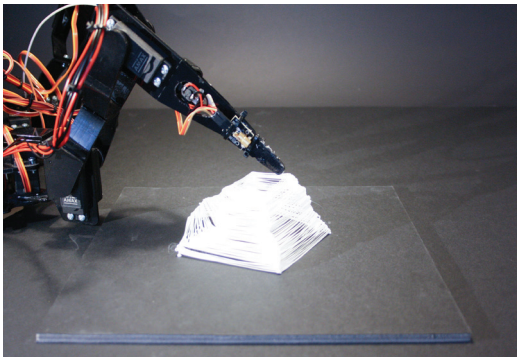
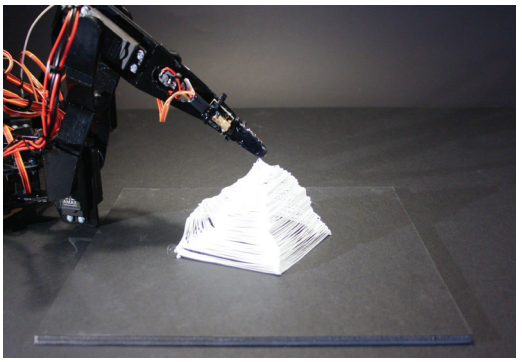
Objekt ID:	PE-12
Material:	PLA 0.75mm
Druckzeit:	15:07 min
Materialmenge:	9g

Abbildung 4.16-24 Printexperiment PE-12 Verlauf

Abbildung 4.16-25 Printexperiment PE-12 Detail

Abbildung 4.16-26 Printexperiment PE-13

Objekt ID:	PE-13
Material:	PLA 0.75mm
Druckzeit:	15:42 min
Materialmenge:	9g

Abbildung 4.16-27 Printexperiment PE-13 Verlauf

Abbildung 4.16-28 Printexperiment PE-13 Detail

5 analyse/synthese

Die Arbeit Bewegt sich im Bereich der Digital Fabrication und Robotik in der Architektur. Zum Vergleich sollen drei Projekte herangezogen werden, welche ähnliche Ziele oder herangeshensweisen verfolgen. Die vorliegende Arbeit verbindet verschiedene Elemente aus den Arbeiten und schafft einen neuen Zugang zur Thematik.

Abbildung 5-1 Adriana Debranova „Sandwright"

Das Projekt „Sandwright" von Adriana Debranova erschafft ein Szenario indem Robotereinheiten mithilfe von Sonnenenergie Wanderdünen verfestigen und so Schattenspendenden Raum in der Wüste schaffen.

Abbildung 5-2 Andrea Graziano „Glitchprint"

Andrea Graziano zeigt mit seinen „Glitchprints" interessante Strukturen aus kleinen Fehlern, sog. Glitches.

Abbildung 5-3 Dave Hakkens „Precious Plastic“

Das Precious Plastic Project von Dave Hakkens aus dem Jahr 2013 zeigt auf, wie einfach Kunstoff recycled und neue Dinge geschaffen werden können. Das Projekt stellt auf OpenSource Basis Pläne für Maschinen zur Verfügung.

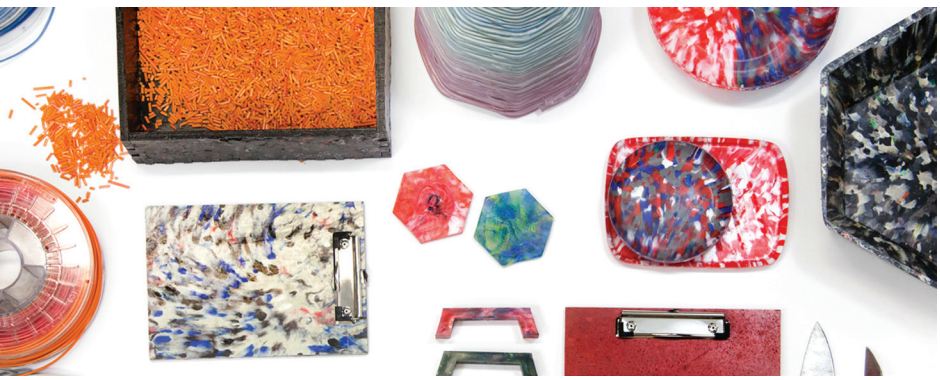
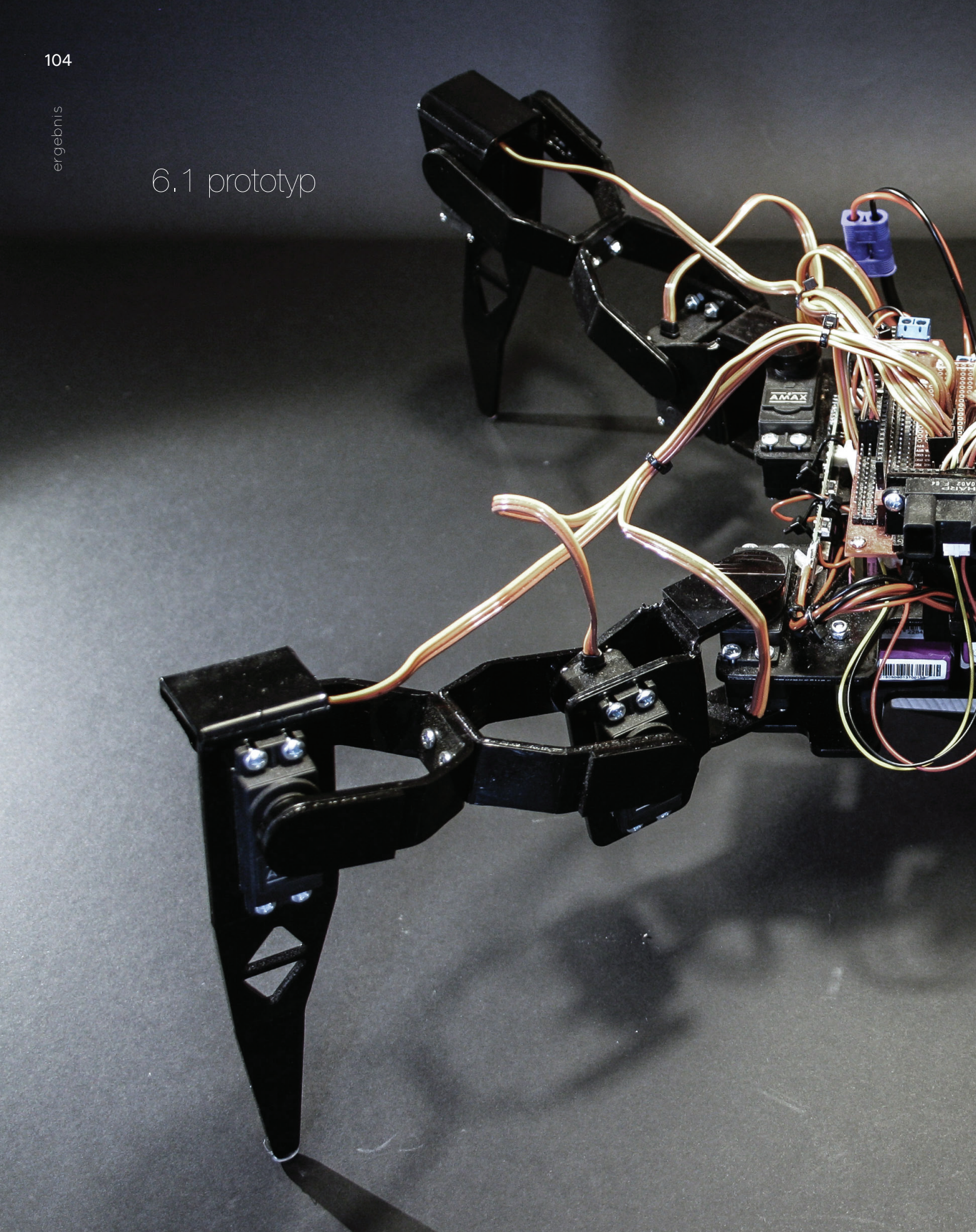


Abbildung 5-4 Dave Hakkens „Precious Plastic“

6 ergebnis

6.1 prototyp



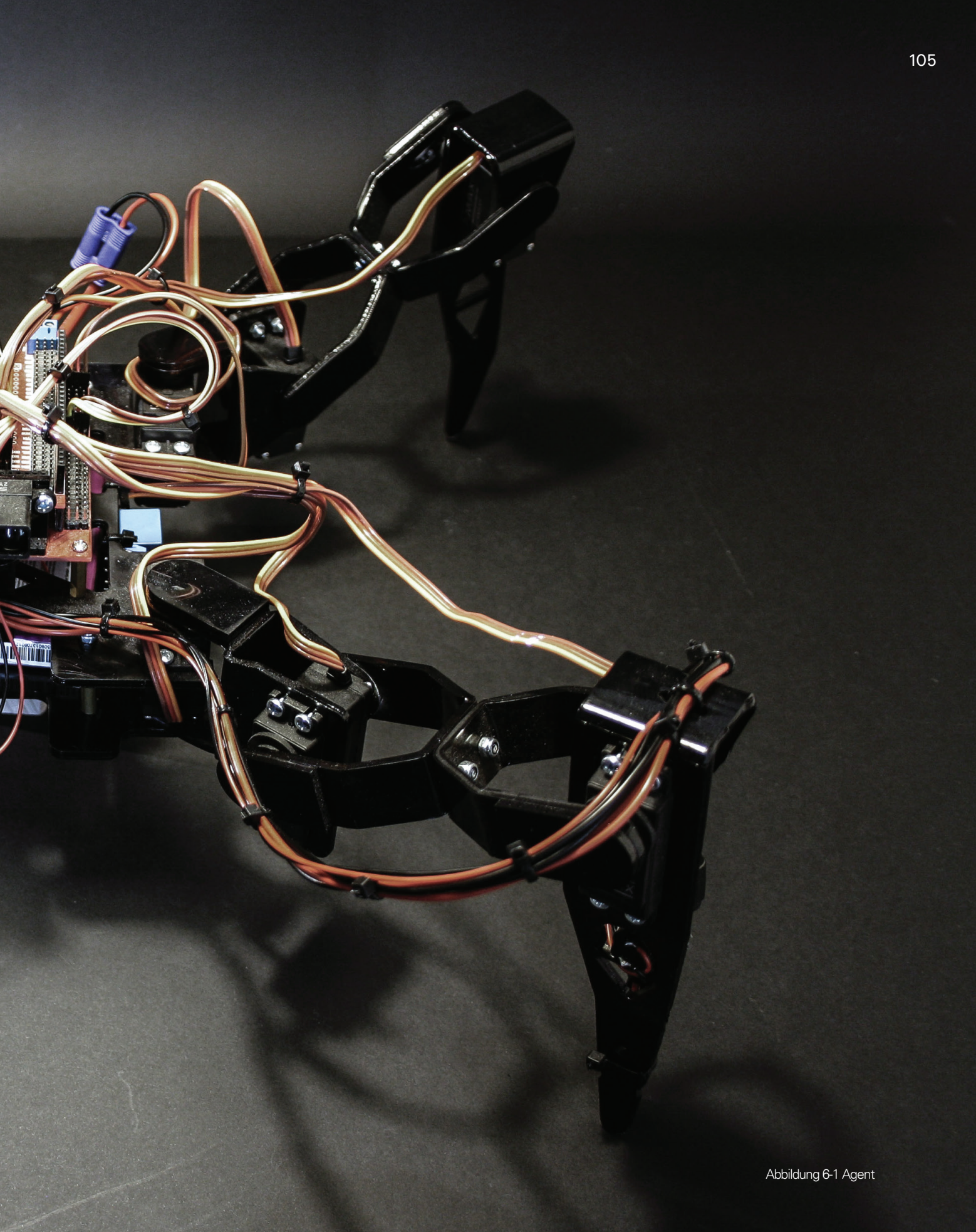
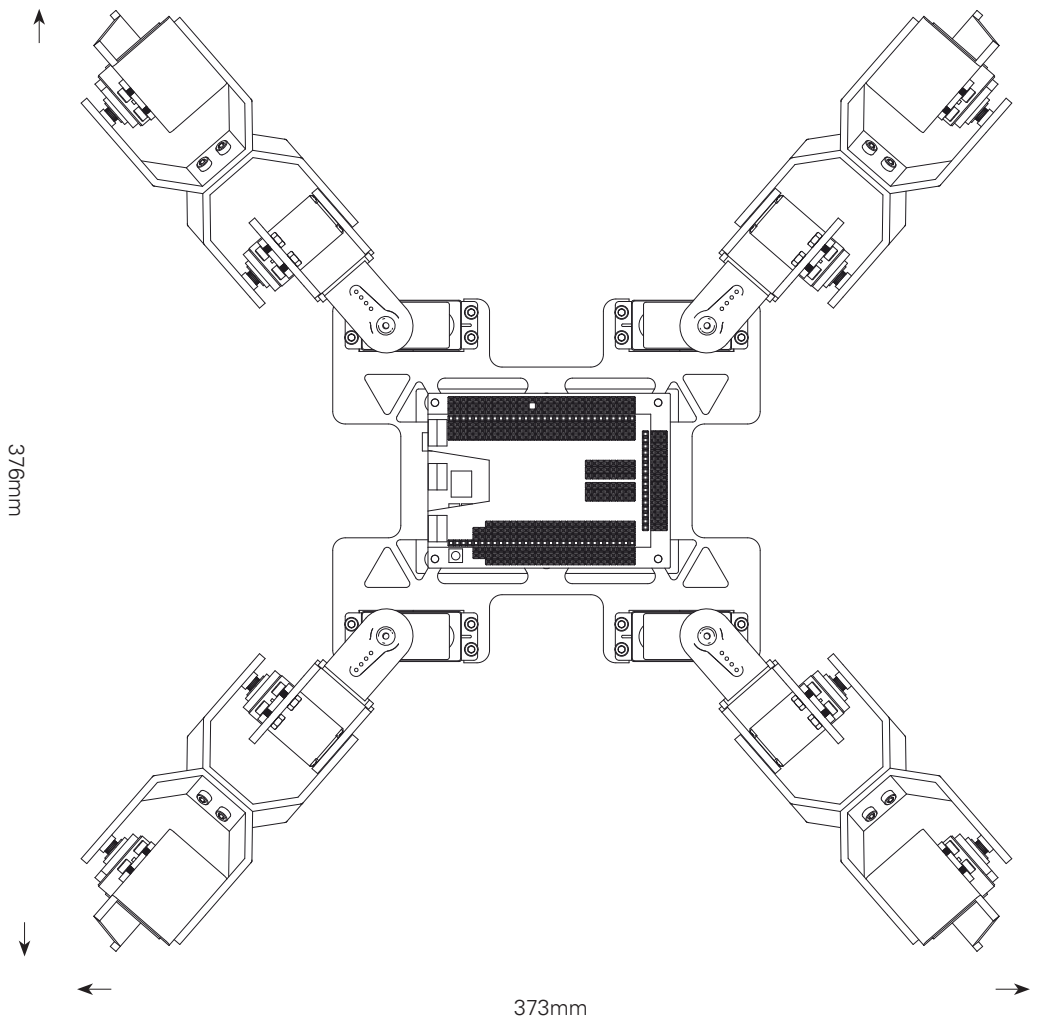
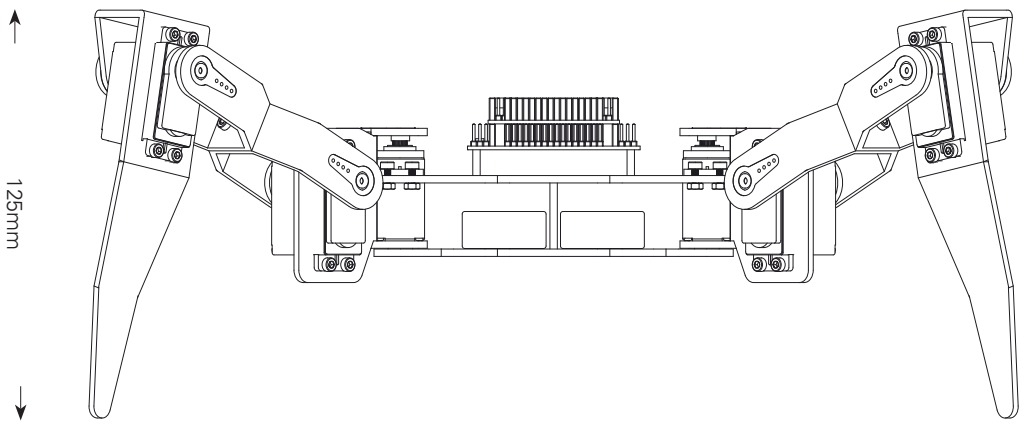


Abbildung 6-1 Agent

6.2 pläne

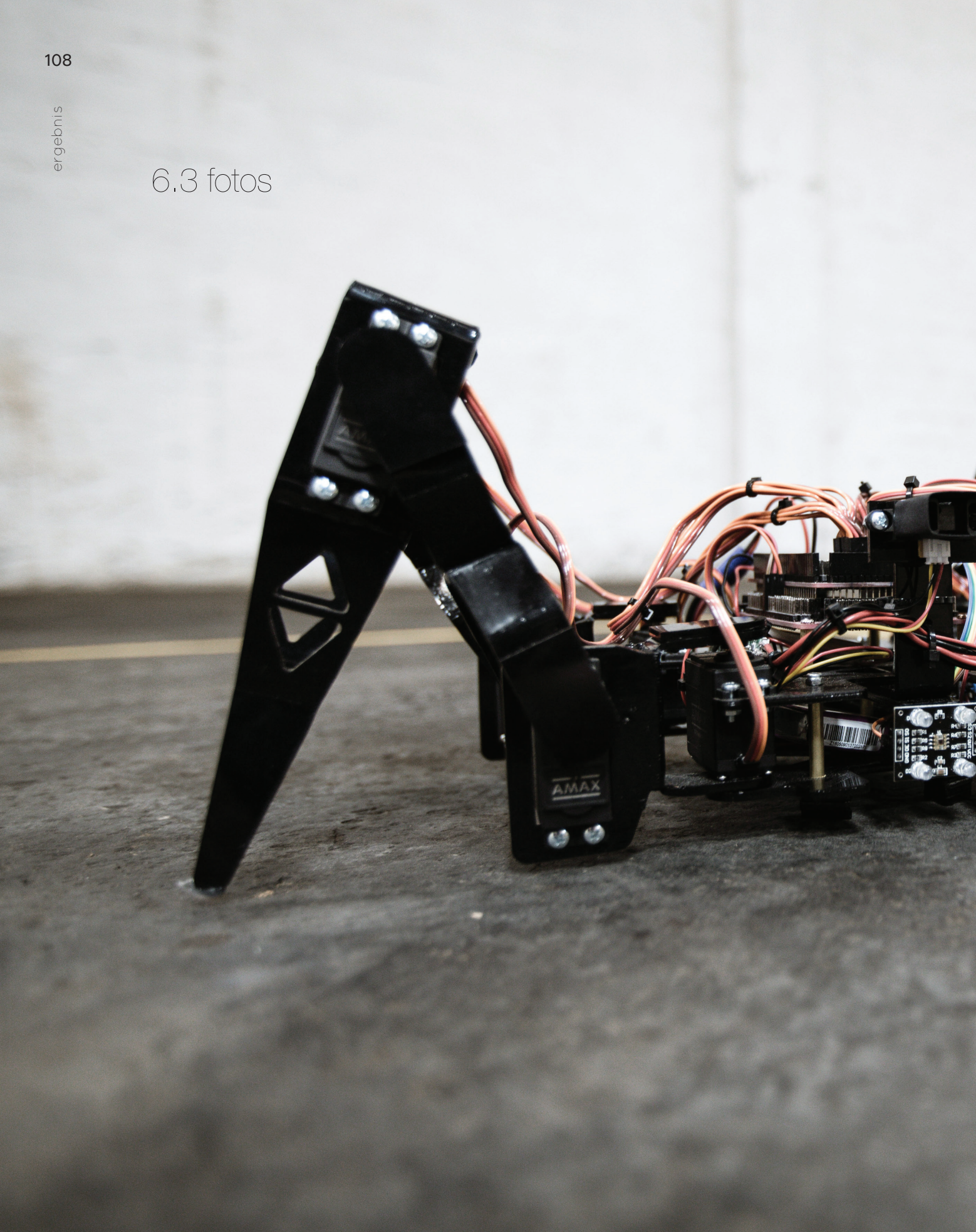


Zeichnung 6.1-1 Draufsicht



Zeichnung 6.1-2 Seitenansicht

6.3 fotos



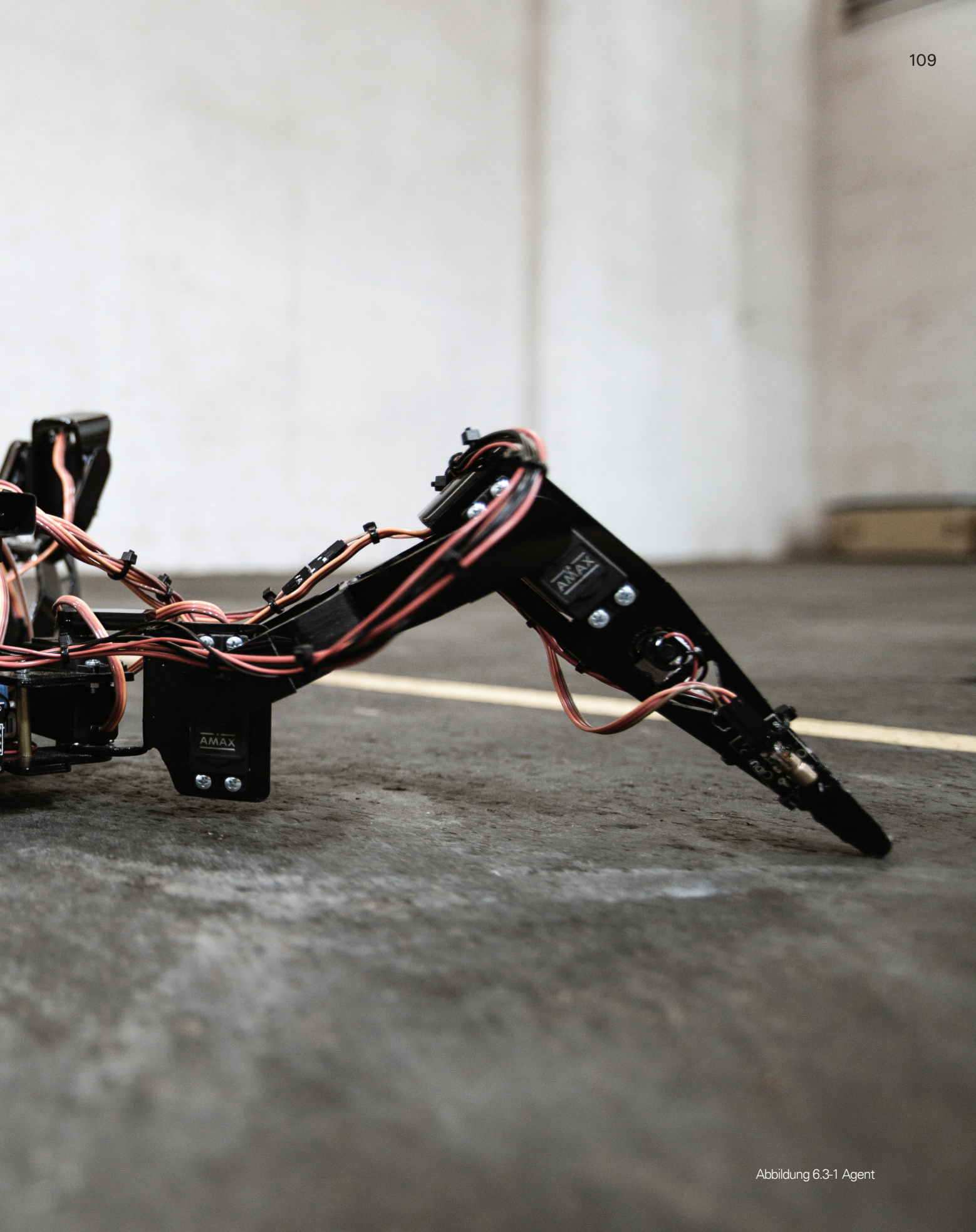
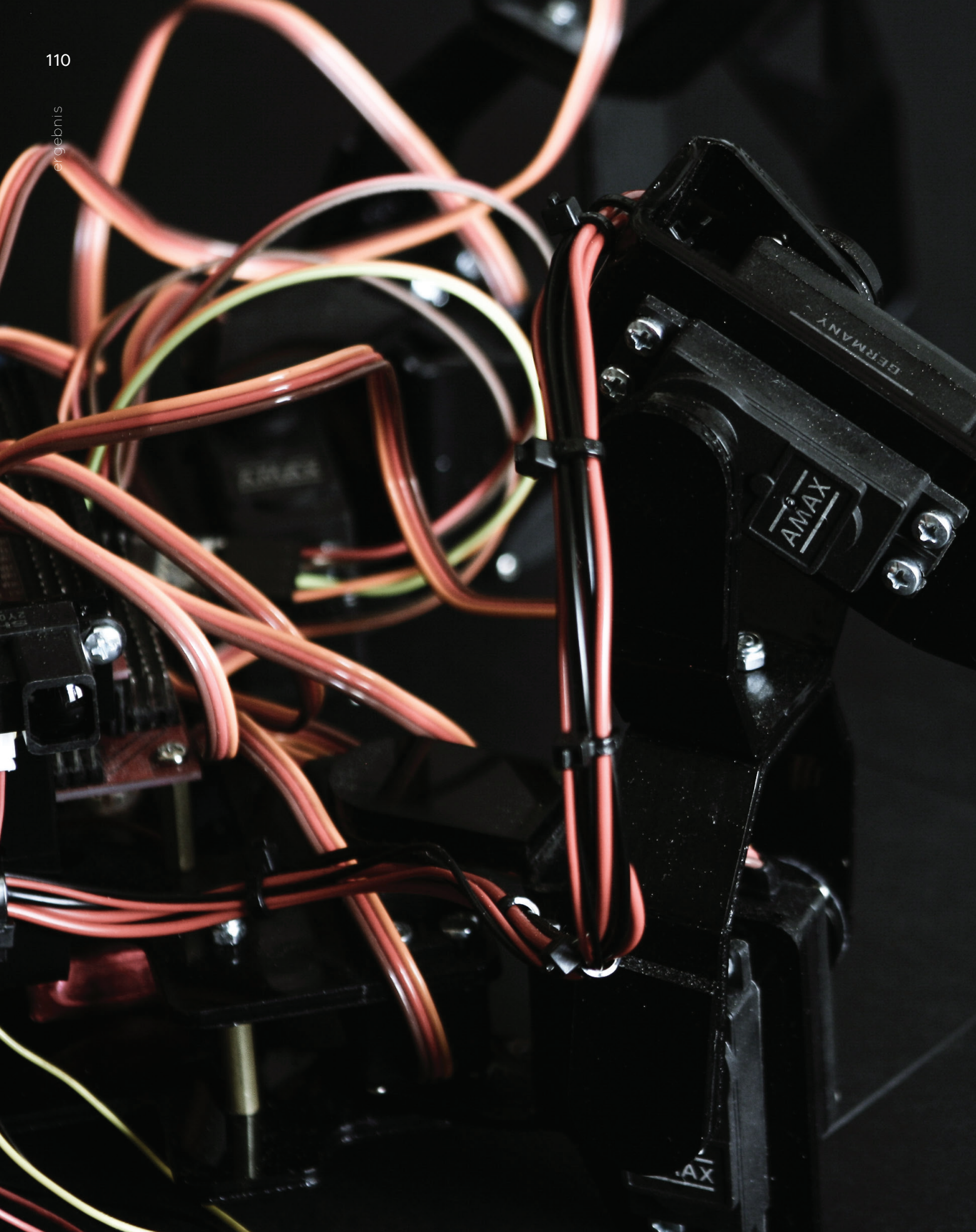


Abbildung 6.3-1 Agent


GERMANY
AMAX

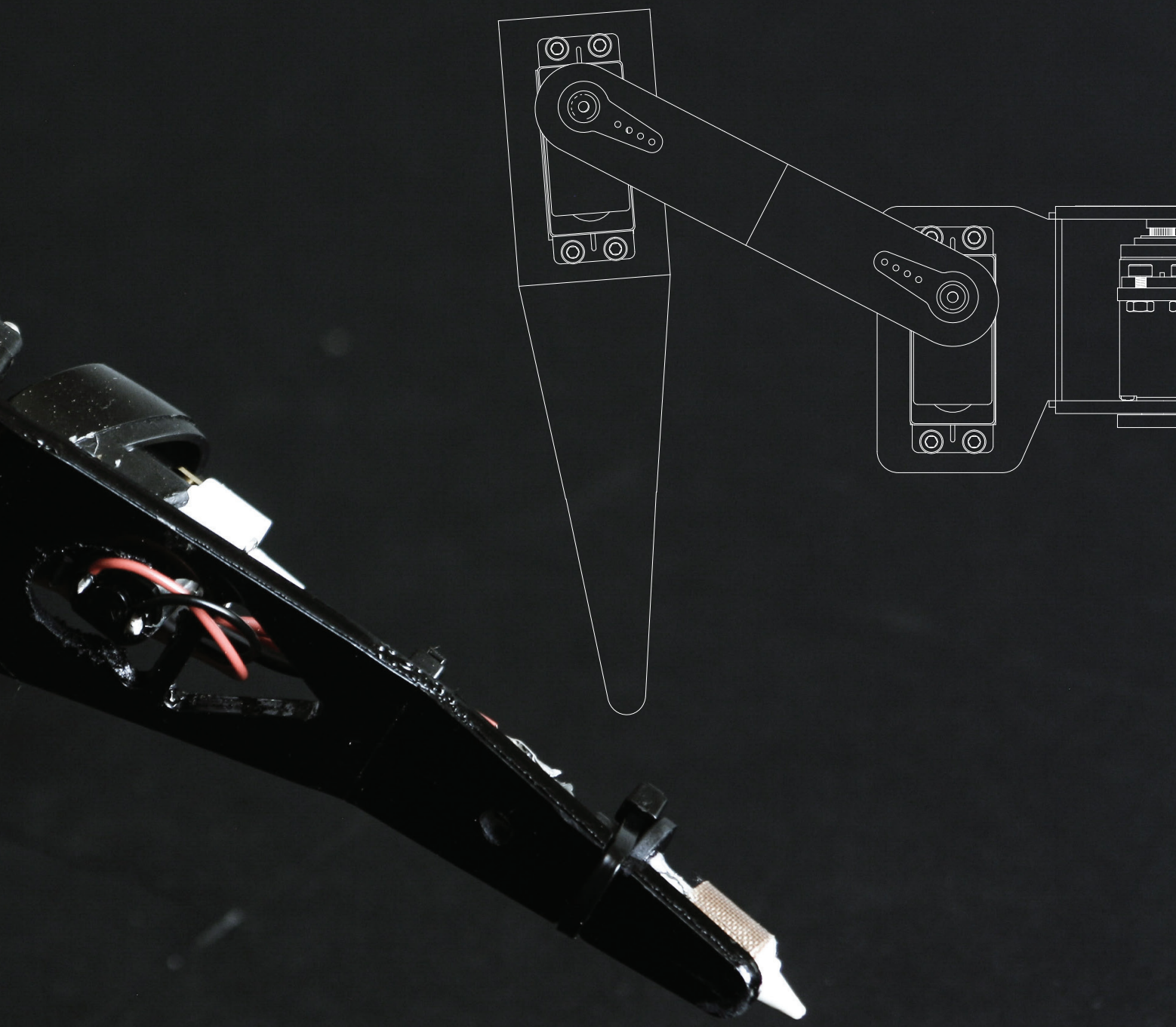


Abbildung 6.3-2 Detail Druckwerkzeug

7 bewertung

Als eine Stärke der vorliegenden Arbeit ist einerseits zu sehen, dass durch die Aufräumarbeiten der autonomen Roboter, natürliche Räume die zuvor weder von Tieren noch von Menschen genutzt werden können eine Transformation erfahren und wieder als Lebensraum zur Verfügung stehen.

Andererseits lässt sich der Begriff der Wiederbenutzbarkeit auch auf die Baumaterialien anwenden. In diesem Fall werden die Materialen nicht eigens hergestellt sondern es werden bereits vorhandene „verlorene" Stoffe (in diesem Fall Kunststoffe aus Plastikmüll) wieder in den Kreislauf zugeführt und als Baumaterialien genutzt.

Eine wichtige Zielsetzung der Arbeit war die Entwicklung eines Roboter-Prototypen welcher in der Lage ist plastische Strukturen zu erstellen.
In längeren Versuchsphasen ist es gelungen einen vierbeinigen Roboter zu entwickeln, welcher sich fortbewegen und ein Bein als Werkzeugarm verwenden kann. Mit ddiesem Werkzeugarm können dreidimensionale Strukturen gedruckt werden.

8 zusammenfassung

Zusammenfassend zeigt die vorliegende Arbeit in Form eines Zukunftsszenarios, wie autonome Maschinen in der Architektur und in der Formgenerierung eingesetzt werden können und gleichzeitig Probleme unserer Zeit adressieren.

Zukünftig könnte das Projekt in verschiedene Richtungen weiterentwicklelt werden.
Mögliche Fragestellungen sind die Skalierung der Roboter, die Weiterentwicklung der Roboter nach technischen sowie ökologischen Gesichtspunkten und die Autonomie der Roboter.

Abbildung 8-1 Bauarbeiter der Zukunft

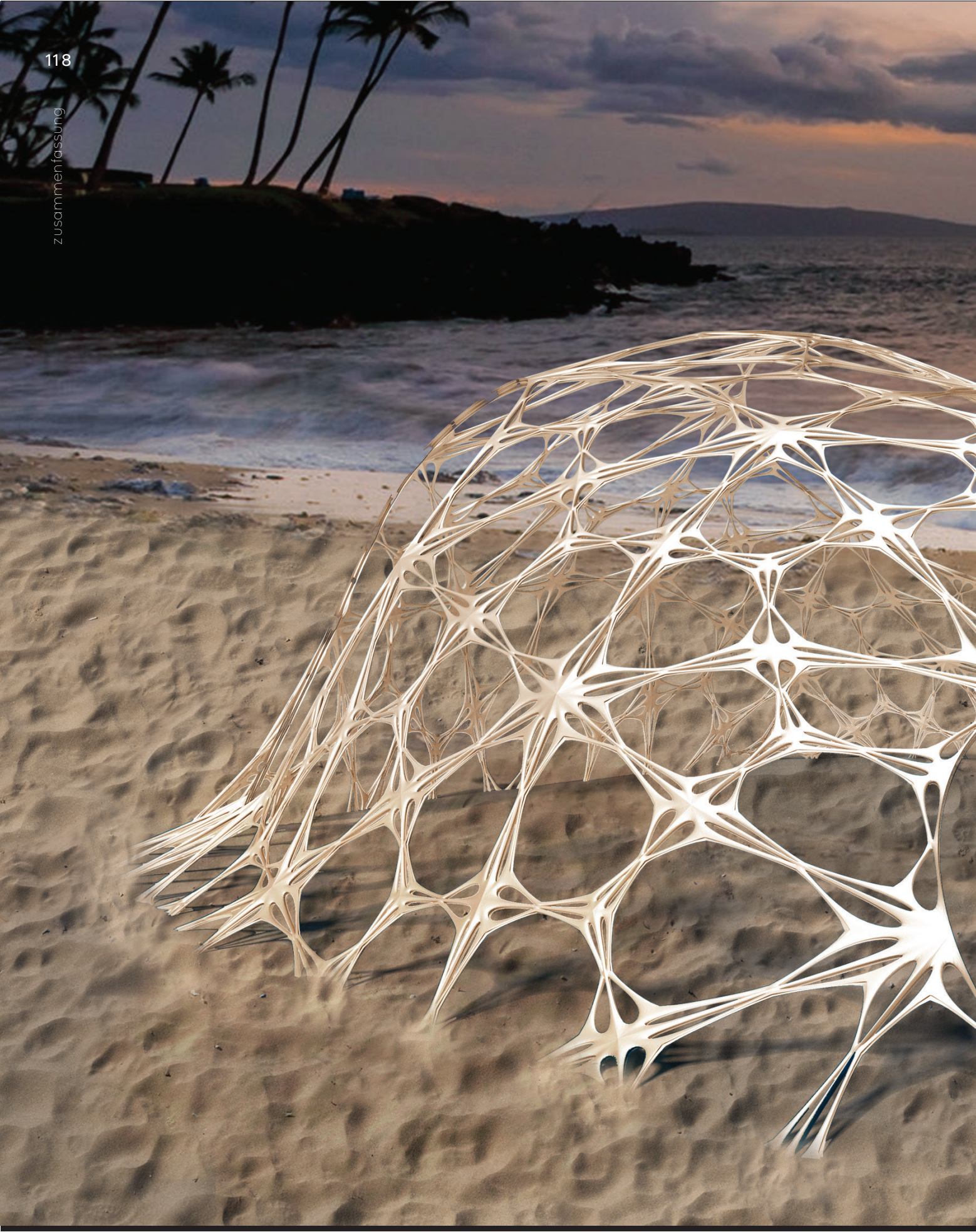


Abbildung 8-2 Mikroarchitektur

9 verzeichnisse

Abbildungsverzeichnis

9	Abbildung 1.1 Verschmutzer Strand	http://www.travelandleisure.com/travel-tips/responsible-travel/hawaii-beach-garbage / Abgerufen am 24.05.2018
13	Diagramm 2.1-1 Linear Economy	The New Plastics Economy, World Economic Forum, 2016
13	Diagramm 2.1-2 Circular Economy	The New Plastics Economy, World Economic Forum, 2016
14	Diagramm 2.1-3 Nicht-recycleter Plastikmüll	http://www.bbc.com/news/science-environment-42264788 Abgerufen am 24.05.2018
15	Diagramm 2.1-4 Globale Kunstoffproduktion	http://www.dw.com/en/six-data-visualizations-that-explain-the-plastic-problem/a-36861883 / Abgerufen am 24.05.2018
18	Zeichnung 2.3-1 Weltkarte mit „Gyren“	Christoph Kolbeck, 2018
19	Zeichnung 2.3-2 Henderson Island	Christoph Kolbeck, 2018
21	Zeichnung 2.3-3 East Beach Henderson Island	Christoph Kolbeck, 2018
26	Zeichnung 4.1-1 Schema Recyclingroboter	Christoph Kolbeck, 2018
26	Abbildung 4.1-1 Schema Konstruktor/Destruktor	Christoph Kolbeck, 2018
27	Zeichnung 4.1-2 Tensile Grid	Christoph Kolbeck, 2018
29	Abbildung 4.1-1 Tensile Grid mit Agenten	Christoph Kolbeck, 2018
30	Abbildung 4.2-1 Experiment zur Materialselbstorganisation	Teamarbeit „Victimless Crime“ Bratislava, 2017
31	Abbildung 4.2-2 Experiment zu Werkzeugpfad	Teamarbeit „Victimless Crime“ Bratislava, 2017
31	Abbildung 4.2-3 Pneu	Teamarbeit „Victimless Crime“ Bratislava, 2017
33	Zeichnung 4.3-1 Schema Bewegung Quadropde	Christoph Kolbeck, 2018
35	Diagramm 4.4-1 Schema Inverskinematik	https://oscarliang.com/inverse-kinematics-and-trigonometry-basics / Abgerufen am 24.05.2018
37	Diagramm 4.5-1 Schema Roboterschaltplan	Christoph Kolbeck, 2018
38	Abbildung 4.6-1 Microcontroller	Christoph Kolbeck, 2018
40	Abbildung 4.7-1 Datenberechnung in Blender	Christoph Kolbeck, 2018
43	Diagramm 4.8-1 Flussdiagramm Roboter	Christoph Kolbeck, 2018
44	Zeichnung 4.9-1 Schema „Verschlaufen“	Christoph Kolbeck, 2018
44	Zeichnung 4.9-2 Schema Formgeneration	Christoph Kolbeck, 2018
47	Zeichnung 4.10-1 Strand vor dem Aussetzen	Christoph Kolbeck, 2018
47	Zeichnung 4.10-3 Sammeln und Strukturen erstellen	Christoph Kolbeck, 2018
47	Zeichnung 4.10-2 Exploremode	Christoph Kolbeck, 2018
47	Zeichnung 4.10-4 Strand mit Strukturen	Christoph Kolbeck, 2018
48	Zeichnung 4.11-1 Tripod Robot	Christoph Kolbeck, 2018
49	Abbildung 4.11-1 Spider Robot	Christoph Kolbeck, 2018
49	Abbildung 4.11-2 Spider Robot	Christoph Kolbeck, 2018
49	Abbildung 4.11-3 Zusammenbau	Christoph Kolbeck, 2018
49	Abbildung 4.11-4 Zusammenbau	Christoph Kolbeck, 2018
50	Abbildung 4.11-5 Detail	Christoph Kolbeck, 2018
51	Abbildung 4.11-6 Agent mit Microcontroller	Christoph Kolbeck, 2018
52	Abbildung 4.11-7 Detail	Christoph Kolbeck, 2018

53	Abbildung 4.11-8 Detail Verbindung	Christoph Kolbeck, 2018
55	Abbildung 4.11-9 Gesamtansicht	Christoph Kolbeck, 2018
56	Abbildung 4.12-1 Schablonen	Christoph Kolbeck, 2018
56	Zeichnung 4.12-1 Einzelteile des Roboters	Christoph Kolbeck, 2018
57	Abbildung 4.12-2 Rückenplatte	Christoph Kolbeck, 2018
57	Abbildung 4.12-4 „Coxa“	Christoph Kolbeck, 2018
57	Abbildung 4.12-3 „Tibia“	Christoph Kolbeck, 2018
57	Abbildung 4.12-5 „Femur“	Christoph Kolbeck, 2018
58	Abbildung 4.13-1 Detail Druckkopf	Christoph Kolbeck, 2018
59	Abbildung 4.13-2 Detail Druckwerkzeug	Christoph Kolbeck, 2018
60	Abbildung 4.14-1 Detail Obstacle Avoidance Sensor	Christoph Kolbeck, 2018
65	Abbildung 4.16-1 Printexperiment PE-01	Christoph Kolbeck, 2018
67	Abbildung 4.16-2 Printexperimente Übersicht	Christoph Kolbeck, 2018
68	Abbildung 4.16-3 Printexperiment PE-01	Christoph Kolbeck, 2018
69	Abbildung 4.16-4 Printexperiment PE-01 Verlauf	Christoph Kolbeck, 2018
70	Abbildung 4.16-5 Printexperiment PE-02	Christoph Kolbeck, 2018
71	Abbildung 4.16-6 Printexperiment PE-02 Verlauf	Christoph Kolbeck, 2018
72	Abbildung 4.16-7 Printexperiment PE-03	Christoph Kolbeck, 2018
73	Abbildung 4.16-8 Printexperiment PE-03 Verlauf	Christoph Kolbeck, 2018
75	Abbildung 4.16-9 Printexperiment PE-04 Detail	Christoph Kolbeck, 2018
77	Abbildung 4.16-10 Printexperiment PE-03	Christoph Kolbeck, 2018
78	Abbildung 4.16-11 Printexperiment PE-04	Christoph Kolbeck, 2018
79	Abbildung 4.16-12 Printexperiment PE-04 Verlauf	Christoph Kolbeck, 2018
80	Abbildung 4.16-13 Printexperiment PE-04	Christoph Kolbeck, 2018
81	Abbildung 4.16-14 Printexperiment PE-04 Verlauf	Christoph Kolbeck, 2018
82	Abbildung 4.16-15 Printexperiment PE-04	Christoph Kolbeck, 2018
83	Abbildung 4.16-16 Printexperiment PE-04 Verlauf	Christoph Kolbeck, 2018
84	Abbildung 4.16-17 Printexperiment PE-09	Christoph Kolbeck, 2018
85	Abbildung 4.16-18 Printexperiment PE-04 Verlauf	Christoph Kolbeck, 2018
87	Abbildung 4.16-19 Printexperiment PE-10 Detail	Christoph Kolbeck, 2018
89	Abbildung 4.16-20 Printexperiment PE-10 Detail	Christoph Kolbeck, 2018
90	Abbildung 4.16-21 Printexperiment PE-11	Christoph Kolbeck, 2018
91	Abbildung 4.16-22 Printexperiment PE-11 Verlauf	Christoph Kolbeck, 2018
92	Abbildung 4.16-23 Printexperiment PE-12	Christoph Kolbeck, 2018
93	Abbildung 4.16-24 Printexperiment PE-12 Verlauf	Christoph Kolbeck, 2018
95	Abbildung 4.16-25 Printexperiment PE-12 Detail	Christoph Kolbeck, 2018
96	Abbildung 4.16-26 Printexperiment PE-13	Christoph Kolbeck, 2018
97	Abbildung 4.16-27 Printexperiment PE-13 Verlauf	Christoph Kolbeck, 2018
99	Abbildung 4.16-28 Printexperiment PE-13 Detail	Christoph Kolbeck, 2018
100	Abbildung 5-1 Adriana Debranova „Sandwright“	https://www.behance.net/gallery/20326693/Sandwright Abgerufen am 24.05.2018

100	Abbildung 5-2 Andrea Graziano „Glitchprint“	http://www.co-de-it.com/wordpress/p5experimental-3d-printing.html / Abgerufen am 24.05.2018
101	Abbildung 5-4 Dave Hakkens „Precious Plastic“	https://preciousplastic.com/en/machines.html Abgerufen am 24.05.2018
101	Abbildung 5-3 Dave Hakkens „Precious Plastic“	https://preciousplastic.com/en/creations.html Abgerufen am 24.05.2018
105	Abbildung 6-1 Agent	Christoph Kolbeck, 2018
106	Zeichnung 6.1-1 Draufsicht	Christoph Kolbeck, 2018
107	Zeichnung 6.1-2 Seitenansicht	Christoph Kolbeck, 2018
109	Abbildung 6.3-1 Agent	Alexander Rieck, 2018
111	Abbildung 6.3-2 Detail Druckwerkzeug	Christoph Kolbeck, 2018
117	Abbildung 8-1 Bauarbeiter der Zukunft	Christoph Kolbeck, 2018
119	Abbildung 8-2 Mikroarchitektur	Christoph Kolbeck, 2018

Scriptverzeichnis

39	Skript 4.6-1 Programmierung Microcontroller, Sprache: C++	Christoph Kolbeck, 2018
41	Skript 4.7-1 Programmierung Microprozessor Sprache: Python	Christoph Kolbeck, 2018
45	Skript 4.9-1 Programmierung Formgeneration, Sprache: Python	Christoph Kolbeck, 2018
61	Skript 4.14-1 Detail Programmierung Obstacle Avoidance Sensor, Sprache: C++	Sharp IR Sensors library for Arduino by Marcal, 2014 https://playground.arduino.cc/Main/SharpIR
124	Spider Robot, Sprache: C++	Regis Hsu, 2015

Tabellenverzeichnis

63	Tabelle 4.15-1 Kunstoffarten und Eigenschaften	Dave Hakkens, Precious Plastic, 2016

Literaturverzeichnis

8 Werner Boote in „Plastic Planet“, 2014

12 The New Plastics Economy - Rethinking the future of plastics, World Economic Forum, 2016

16 Detail - Boom der Robotik, 2017
https://www.detail.de/artikel/boom-der-robotik-29660/
Abgerufen am 24.05.2018

16 Donna Haraway, Companion Species Manifesto, 2003

18 UNESCO - Henderson Island
https://whc.unesco.org/en/list/487
Abgerufen am 24.05.2018

32 Robot Platform, 2013
http://www.robotplatform.com/knowledge/Classification_of_Robots/legged_robots.html
Abgerufen am 24.05.2018

34 Oscar Liang, 2012
https://oscarliang.com/inverse-kinematics-and-trigonometry-basics/
Abgerufen am 24.05.2018

48 Regis Hsu, 2015
http://regishsu.blogspot.co.at/search/label/0.SpiderRobot
Abgerufen am 24.05.2018

62 Reprap, 2014
http://reprap.org/wiki/ABS_HDPE_blend
Abgerufen am 24.05.2018

3D-Modellverzeichnis

48 Spider Robot, Regis Hsu
http://regishsu.blogspot.co.at/search/label/0.SpiderRobot
Abgerufen am 24.05.2018

Verwendete Software

Modeler: Rhinoceros
Programmierung: Arduino IDE
Digital Content Creator: Blender
Renderer: Keyshot, Blender

9.1 code

KALIBRIEREN

```
// Locate the initial position of legs
// RegisHsu 2015-09-09
// Modified by Chris Kolbeck 2018-02-20

#include <Servo.h>

Servo servo[4][3];

//define servos' ports
const int servo_pin[4][3] = { {30, 31, 32}, {34, 35, 36}, {39, 40, 41}, {43, 44, 45} };

void setup()
{
  //initialize all servos
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].attach(servo_pin[i][j]);
      delay(20);
    }
  }
}

void loop(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].write(90);
      delay(20);
      }
   }
}
```

```
/* ----------------------------------------------------------------------------
  - Project: Remote control Crawling robot
  - Author:  panerqiang@sunfounder.com
  - Date:  2015/1/27

// modified by Regis for spider project, 2015-09-26
// add remote control by HC-06 bluetooth module
// modified by Chris Kolbeck for WeaverBot with Blender controls

int active = 0; // Variable als Schalter um zu erkennen, das die Befehle von Blender kommen
int blender_control_servo_1; // Wert von Servo 1
int blender_control_servo_2; // Wert von Servo 2
int blender_control_servo_3; // Wert von Servo 3
const int relaisPin = 10;
const int laserPin = 9;

// For Sharp IR Distance

#include <SharpIR.h>

#define ir A10
#define model 20150
// ir: the pin where your sensor is attached
// model: an int that determines your sensor:  1080 for GP2Y0A21Y
//                                             20150 for GP2Y0A02Y
//                                             (working distance range according to the datasheets)

SharpIR SharpIR(ir, model);
long randNumber;
//int dis=SharpIR.distance();
//static step = 0;

/* Includes ------------------------------------------------------------------*/
#include <Servo.h>    //to define and control servos
#include <FlexiTimer2.h>//to set a timer to manage all servos

/* Servos --------------------------------------------------------------------*/
//define 12 servos for 4 legs
Servo servo[4][3];
//define servos' ports
const int servo_pin[4][3] = { {30, 31, 32}, {34, 35, 36}, {39, 40, 41}, {43, 44, 45} };
/* Size of the robot ---------------------------------------------------------*/
```

```
const float length_a = 55;
const float length_b = 77.5;
const float length_c = 27.5;
const float length_side = 71;
const float z_absolute = -25;
//const float z_absolute = -35;
/* Constants for movement ------------------------------------------------------*/
const float z_default = -50, z_up = -30, z_boot = z_absolute;
const float x_default = 62, x_offset = 0;
const float y_start = 0, y_step = 40;
/* variables for movement ------------------------------------------------------*/
volatile float site_now[4][3];    //real-time coordinates of the end of each leg
volatile float site_expect[4][3]; //expected coordinates of the end of each leg
float temp_speed[4][3];   //each axis' speed, needs to be recalculated before each movement
float move_speed;     //movement speed
float speed_multiple = 1; //movement speed multiple
const float spot_turn_speed = 4;
const float leg_move_speed = 8;
const float body_move_speed = 3;
const float stand_seat_speed = 1;
volatile int rest_counter;      //+1/0.02s, for automatic rest
//functions' parameter
const float KEEP = 255;
//define PI for calculation
const float pi = 3.1415926;
/* Constants for turn ----------------------------------------------------------*/
//temp length
const float temp_a = sqrt(pow(2 * x_default + length_side, 2) + pow(y_step, 2));
const float temp_b = 2 * (y_start + y_step) + length_side;
const float temp_c = sqrt(pow(2 * x_default + length_side, 2) + pow(2 * y_start + y_step + length_
side, 2));
const float temp_alpha = acos((pow(temp_a, 2) + pow(temp_b, 2) - pow(temp_c, 2)) / 2 / temp_a /
temp_b);
//site for turn
const float turn_x1 = (temp_a - length_side) / 2;
const float turn_y1 = y_start + y_step / 2;
const float turn_x0 = turn_x1 - temp_b * cos(temp_alpha);
const float turn_y0 = temp_b * sin(temp_alpha) - turn_y1 - length_side;
/* ---------------------------------------------------------------------------*/

/*
  - setup function
```

```
   ----------------------------------------------------------------------------*/
void setup()
{
  //start serial for debug
  Serial.begin(9600);
  Serial.println("Robot starts initialization");

  //initialize default parameter
  set_site(0, x_default - x_offset, y_start + y_step, z_boot);
  set_site(1, x_default - x_offset, y_start + y_step, z_boot);
  set_site(2, x_default + x_offset, y_start, z_boot);
  set_site(3, x_default + x_offset, y_start, z_boot);
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      site_now[i][j] = site_expect[i][j];
    }
  }
  //start servo service
  FlexiTimer2::set(20, servo_service);
  FlexiTimer2::start();
  Serial.println("Servo service started");
  //initialize servos
  servo_attach();
  Serial.println("Servos initialized");
  pinMode(relaisPin, OUTPUT); // Printer relais
  pinMode(laserPin, OUTPUT); // Laser
  Serial.println("Printer initialized");
  Serial.println("Robot initialization Complete");

  delay(15000); // Wartezeit bevor Roboter startet
  stand();
  delay(2000);
  hand_wave(3);
  delay(2000);
}


void servo_attach(void)
{
  for (int i = 0; i < 4; i++)
```

```
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].attach(servo_pin[i][j]);
      delay(100);
    }
  }
}

void servo_detach(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].detach();
      delay(100);
    }
  }
}

/*
  - robot operation function/loop function
   ------------------------------------------------------------------------------*/
void loop(void) {

  if (Serial.available() > 0) {

    int receiving = 1;

    if (receiving == 1) {

      int action = Serial.parseInt();

      // to print
      if (action == 1) {
        printer();
      }

      // to stop flexitimer
      if (action == 2) {
        digitalWrite(laserPin, HIGH);
```

```
  delay(15);
  FlexiTimer2::stop();
  delay(15);
}

// to stand
if (action == 5) {
  stand();
  delay(2000);
}

// to sit
if (action == 6) {
  sit();
  delay(100);
}

// to default position
if (action == 7) {
  default_position();
  delay(2000);
}


// to recieve from blender
if (action == 8) {

  // empfangen

  blender_control_servo_1 = Serial.parseInt();
  blender_control_servo_2 = Serial.parseInt();
  blender_control_servo_3 = Serial.parseInt();

  // position anfahren

  servo[1][2].write(blender_control_servo_1);
  delay(15);
  servo[1][0].write(blender_control_servo_2);
  delay(15);
  servo[1][1].write(blender_control_servo_3);
  delay(15);
```

```
      }

      if (action == 9) {
        FlexiTimer2::start();
        delay(15);
        digitalWrite(laserPin, LOW);
        delay(15);
        default_position();
        delay(2000);
        stand();
        delay(2000);
        step_forward(3);
        turn_right(5);
        receiving = 0;
        delay(2000);


      }
    }
  }

  else {
    explore();
  }
}



/*
  - printer switch
  -----------------------------------------------------------------------------*/
void printer(void)
{
  digitalWrite(relaisPin, HIGH);
  delay(500);
  digitalWrite(relaisPin, LOW);
  delay(2000);
}

/*
  - explore
```

```
   ----------------------------------------------------------------------------*/
void explore(void) {

  int dis = SharpIR.distance();

  if (dis < 40) {
    step_forward(3);
    delay(1500);

    randNumber = random(1, 3);

    if (randNumber == 1) {
      turn_right(5);
      delay(2000);
    }

    else if (randNumber == 2) {
      turn_left(5);
      delay(2000);
    }
  }

  else {
    step_back(1);
  }
}



void default_position(void) {

  set_site(0, x_default - x_offset, y_start + y_step, z_boot);
  set_site(1, x_default - x_offset, y_start + y_step, z_boot);
  set_site(2, x_default + x_offset, y_start, z_boot);
  set_site(3, x_default + x_offset, y_start, z_boot);
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      site_now[i][j] = site_expect[i][j];
    }
  }
```

```
}



/*
  - is_stand
   ------------------------------------------------------------------------*/
bool is_stand(void)
{
  if (site_now[0][2] == z_default)
    return true;
  else
    return false;
}

/*
  - sit
  - blocking function
   ------------------------------------------------------------------------*/
void sit(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_boot);
  }
  wait_all_reach();
}



/*
  - stand
  - blocking function
   ------------------------------------------------------------------------*/
void stand(void)
{
  move_speed = stand_seat_speed;
  for (int leg = 0; leg < 4; leg++)
  {
    set_site(leg, KEEP, KEEP, z_default);
  }
  wait_all_reach();
```

```
}


/*
  - spot turn to left
  - blocking function
  - parameter step steps wanted to turn
   ------------------------------------------------------------------------------*/
void turn_left(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[3][1] == y_start)
    {
      //leg 3&1 move
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x1 - x_offset, turn_y1, z_default);
      set_site(1, turn_x0 - x_offset, turn_y0, z_default);
      set_site(2, turn_x1 + x_offset, turn_y1, z_default);
      set_site(3, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(3, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x1 + x_offset, turn_y1, z_default);
      set_site(1, turn_x0 + x_offset, turn_y0, z_default);
      set_site(2, turn_x1 - x_offset, turn_y1, z_default);
      set_site(3, turn_x0 - x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(1, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default + x_offset, y_start, z_default);
      set_site(1, x_default + x_offset, y_start, z_up);
      set_site(2, x_default - x_offset, y_start + y_step, z_default);
      set_site(3, x_default - x_offset, y_start + y_step, z_default);
      wait_all_reach();
```

```
      set_site(1, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
    else
    {
      //leg 0&2 move
      set_site(0, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_up);
      set_site(1, turn_x1 + x_offset, turn_y1, z_default);
      set_site(2, turn_x0 - x_offset, turn_y0, z_default);
      set_site(3, turn_x1 - x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x0 - x_offset, turn_y0, z_default);
      set_site(1, turn_x1 - x_offset, turn_y1, z_default);
      set_site(2, turn_x0 + x_offset, turn_y0, z_default);
      set_site(3, turn_x1 + x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(2, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_up);
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();

      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - spot turn to right
```

```
  - blocking function
  - parameter step steps wanted to turn
   ------------------------------------------------------------------------*/
void turn_right(unsigned int step)
{
  move_speed = spot_turn_speed;
  while (step-- > 0)
  {
    if (site_now[2][1] == y_start)
    {
      //leg 2&0 move
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x0 - x_offset, turn_y0, z_default);
      set_site(1, turn_x1 - x_offset, turn_y1, z_default);
      set_site(2, turn_x0 + x_offset, turn_y0, z_up);
      set_site(3, turn_x1 + x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(2, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_default);
      set_site(1, turn_x1 + x_offset, turn_y1, z_default);
      set_site(2, turn_x0 - x_offset, turn_y0, z_default);
      set_site(3, turn_x1 - x_offset, turn_y1, z_default);
      wait_all_reach();

      set_site(0, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default + x_offset, y_start, z_up);
      set_site(1, x_default + x_offset, y_start, z_default);
      set_site(2, x_default - x_offset, y_start + y_step, z_default);
      set_site(3, x_default - x_offset, y_start + y_step, z_default);
      wait_all_reach();

      set_site(0, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
    else
```

```
    {
      //leg 1&3 move
      set_site(1, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(0, turn_x1 + x_offset, turn_y1, z_default);
      set_site(1, turn_x0 + x_offset, turn_y0, z_up);
      set_site(2, turn_x1 - x_offset, turn_y1, z_default);
      set_site(3, turn_x0 - x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(1, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(0, turn_x1 - x_offset, turn_y1, z_default);
      set_site(1, turn_x0 - x_offset, turn_y0, z_default);
      set_site(2, turn_x1 + x_offset, turn_y1, z_default);
      set_site(3, turn_x0 + x_offset, turn_y0, z_default);
      wait_all_reach();

      set_site(3, turn_x0 + x_offset, turn_y0, z_up);
      wait_all_reach();

      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_default);
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();

      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - go forward
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------------*/
void step_forward(unsigned int step)
{
```

```
move_speed = leg_move_speed;
while (step-- > 0)
{
  if (site_now[2][1] == y_start)
  {
    //leg 2&1 move
    set_site(2, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = body_move_speed;

    set_site(0, x_default + x_offset, y_start, z_default);
    set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
    set_site(2, x_default - x_offset, y_start + y_step, z_default);
    set_site(3, x_default - x_offset, y_start + y_step, z_default);
    wait_all_reach();

    move_speed = leg_move_speed;

    set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(1, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(1, x_default + x_offset, y_start, z_default);
    wait_all_reach();
  }
  else
  {
    //leg 0&3 move
    set_site(0, x_default + x_offset, y_start, z_up);
    wait_all_reach();
    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
    wait_all_reach();
    set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
    wait_all_reach();

    move_speed = body_move_speed;
```

```
      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start, z_default);
      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();

      move_speed = leg_move_speed;

      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}

/*
  - go back
  - blocking function
  - parameter step steps wanted to go
   ----------------------------------------------------------------------------*/
void step_back(unsigned int step)
{
  move_speed = leg_move_speed;
  while (step-- > 0)
  {
    if (site_now[3][1] == y_start)
    {
      //leg 3&0 move
      set_site(3, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(3, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();

      move_speed = body_move_speed;

      set_site(0, x_default + x_offset, y_start + 2 * y_step, z_default);
      set_site(1, x_default + x_offset, y_start, z_default);
```

```
      set_site(2, x_default - x_offset, y_start + y_step, z_default);
      set_site(3, x_default - x_offset, y_start + y_step, z_default);
      wait_all_reach();

      move_speed = leg_move_speed;

      set_site(0, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(0, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(0, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
    else
    {
      //leg 1&2 move
      set_site(1, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(1, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(1, x_default + x_offset, y_start + 2 * y_step, z_default);
      wait_all_reach();

      move_speed = body_move_speed;

      set_site(0, x_default - x_offset, y_start + y_step, z_default);
      set_site(1, x_default - x_offset, y_start + y_step, z_default);
      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_default);
      set_site(3, x_default + x_offset, y_start, z_default);
      wait_all_reach();

      move_speed = leg_move_speed;

      set_site(2, x_default + x_offset, y_start + 2 * y_step, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_up);
      wait_all_reach();
      set_site(2, x_default + x_offset, y_start, z_default);
      wait_all_reach();
    }
  }
}
```

```
// add by RegisHsu

void body_left(int i)
{
  set_site(0, site_now[0][0] + i, KEEP, KEEP);
  set_site(1, site_now[1][0] + i, KEEP, KEEP);
  set_site(2, site_now[2][0] - i, KEEP, KEEP);
  set_site(3, site_now[3][0] - i, KEEP, KEEP);
  wait_all_reach();
}

void body_right(int i)
{
  set_site(0, site_now[0][0] - i, KEEP, KEEP);
  set_site(1, site_now[1][0] - i, KEEP, KEEP);
  set_site(2, site_now[2][0] + i, KEEP, KEEP);
  set_site(3, site_now[3][0] + i, KEEP, KEEP);
  wait_all_reach();
}

void hand_wave(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(2, turn_x1, turn_y1, 50);
      wait_all_reach();
      set_site(2, turn_x0, turn_y0, 50);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
```

```
    wait_all_reach();
    move_speed = 1;
    body_left(15);
  }
  else
  {
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, turn_x1, turn_y1, 50);
      wait_all_reach();
      set_site(0, turn_x0, turn_y0, 50);
      wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}

void hand_shake(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(2, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
```

```
      set_site(2, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
  }
  else
  {
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(0, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}

void spray(int i)
{
  float x_tmp;
  float y_tmp;
  float z_tmp;
  move_speed = 1;
  if (site_now[3][1] == y_start)
  {
    body_right(15);
    x_tmp = site_now[2][0];
    y_tmp = site_now[2][1];
    z_tmp = site_now[2][2];
    move_speed = body_move_speed;
```

```
    for (int j = 0; j < i; j++)
    {
      set_site(2, turn_x1, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(2, turn_x0, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(2, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_left(15);
  }
  else
  {
    body_left(15);
    x_tmp = site_now[0][0];
    y_tmp = site_now[0][1];
    z_tmp = site_now[0][2];
    move_speed = body_move_speed;
    for (int j = 0; j < i; j++)
    {
      set_site(0, x_default - 30, y_start + 2 * y_step, 55);
      wait_all_reach();
      set_site(0, x_default - 30, y_start + 2 * y_step, 10);
      wait_all_reach();
    }
    set_site(0, x_tmp, y_tmp, z_tmp);
    wait_all_reach();
    move_speed = 1;
    body_right(15);
  }
}

/*
  - microservos service /timer interrupt function/50Hz
  - when set site expected,this function move the end point to it in a straight line
  - temp_speed[4][3] should be set before set expect site,it make sure the end point
   move in a straight line,and decide move speed.
   ------------------------------------------------------------------------------*/
void servo_service(void)
{
  sei();
```

```
  static float alpha, beta, gamma;

  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      if (abs(site_now[i][j] - site_expect[i][j]) >= abs(temp_speed[i][j]))
        site_now[i][j] += temp_speed[i][j];
      else
        site_now[i][j] = site_expect[i][j];
    }

    cartesian_to_polar(alpha, beta, gamma, site_now[i][0], site_now[i][1], site_now[i][2]);
    polar_to_servo(i, alpha, beta, gamma);
  }

  rest_counter++;
}

/*
  - set one of end points' expect site
  - this founction will set temp_speed[4][3] at same time
  - non - blocking function
   ---------------------------------------------------------------------------*/
void set_site(int leg, float x, float y, float z)
{
  float length_x = 0, length_y = 0, length_z = 0;

  if (x != KEEP)
    length_x = x - site_now[leg][0];
  if (y != KEEP)
    length_y = y - site_now[leg][1];
  if (z != KEEP)
    length_z = z - site_now[leg][2];

  float length = sqrt(pow(length_x, 2) + pow(length_y, 2) + pow(length_z, 2));

  temp_speed[leg][0] = length_x / length * move_speed * speed_multiple;
  temp_speed[leg][1] = length_y / length * move_speed * speed_multiple;
  temp_speed[leg][2] = length_z / length * move_speed * speed_multiple;

  if (x != KEEP)
```

```
    site_expect[leg][0] = x;
  if (y != KEEP)
    site_expect[leg][1] = y;
  if (z != KEEP)
    site_expect[leg][2] = z;
}

/*
  - wait one of end points move to expect site
  - blocking function
   ----------------------------------------------------------------------------*/
void wait_reach(int leg)
{
  while (1)
    if (site_now[leg][0] == site_expect[leg][0])
      if (site_now[leg][1] == site_expect[leg][1])
        if (site_now[leg][2] == site_expect[leg][2])
          break;
}

/*
  - wait all of end points move to expect site
  - blocking function
   ----------------------------------------------------------------------------*/
void wait_all_reach(void)
{
  for (int i = 0; i < 4; i++)
    wait_reach(i);
}

/*
  - trans site from cartesian to polar
  - mathematical model 2/2
   ----------------------------------------------------------------------------*/
void cartesian_to_polar(volatile float & alpha, volatile float & beta, volatile float & gamma,
volatile float x, volatile float y, volatile float z)
{
  //calculate w-z degree
  float v, w;
  w = (x >= 0 ? 1 : -1) * (sqrt(pow(x, 2) + pow(y, 2)));
  v = w - length_c;
  alpha = atan2(z, v) + acos((pow(length_a, 2) - pow(length_b, 2) + pow(v, 2) + pow(z, 2)) / 2 /
```

```
length_a / sqrt(pow(v, 2) + pow(z, 2)));
  beta = acos((pow(length_a, 2) + pow(length_b, 2) - pow(v, 2) - pow(z, 2)) / 2 / length_a /
length_b);
  //calculate x-y-z degree
  gamma = (w >= 0) ? atan2(y, x) : atan2(-y, -x);
  //trans degree pi->180
  alpha = alpha / pi * 180;
  beta = beta / pi * 180;
  gamma = gamma / pi * 180;
}

/*
  - trans site from polar to microservos
  - mathematical model map to fact
  - the errors saved in eeprom will be add
   ---------------------------------------------------------------------------*/
void polar_to_servo(int leg, float alpha, float beta, float gamma)
{
  if (leg == 0)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }
  else if (leg == 1)
  {
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 2)
  {
    alpha += 90;
    beta = 180 - beta;
    gamma = 90 - gamma;
  }
  else if (leg == 3)
  {
    alpha = 90 - alpha;
    beta = beta;
    gamma += 90;
  }

  servo[leg][0].write(alpha);
```

```
  servo[leg][1].write(beta);
  servo[leg][2].write(gamma);
}
```

10 curriculum vitae

Christoph Kolbeck

Geboren in Landshut, Niederbayern 1987

Bildung

10/2009-07/2013	Hochschule Rosenheim Studium der Innenarchitektur Bachelor of Arts Innenarchitektur
10/2013-03/2016	Technische Universität Wien Studium der Architektur Bachelor of Science Architektur
04/2016-06/2018	Technische Universität Wien Studium der Architektur Master of Science Architektur/Diplom-Ingenieur

Berufserfahrung (Praktika/Projekte)

08/2010-09/2011	Holzrausch GmbH, Forstern b. München Praktikant im Raumbildenenden Ausbau
10/2011-02/2012	Feigel-Huber Architekten, Landshut Praktikant in der Planung
03/2012-09/2013	Hochschule Rosenheim Forschung und Entwicklung Mitarbeit im AUDI Imagine FuE Projekt in Kooperation mit der Audi AG Ingolstadt

11 danksagung

An dieser Stelle möchte ich mich herzlich bei allen bedanken die diese Arbeit möglich gemacht und mich dabei unterstützt haben.

Ein besonderer Dank geht an meine Eltern für die Unterstützung im Studium.

Danke an Alex und Jonathan für die Hilfe beim Filmen.
Bei Lena für die Gespräche und die Motivation.

Und nicht zuletzt bei meinen Betreuern Manfred Berthold und Christoph Müller für die außerordentliche Betreuung.

Danke.