Creating an obstacle sensor for robotic tasks

BY Max Schaufelberger

RESEARCH & DEVELOPMENT PROGRAMME

SS2017

DDU
Digital Design Unit — Digitales Gestalten

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Creating an obstacle sensor for robotic tasks
BY Max Schaufelberger

PROJECT SUPERVISOR
DIPL.-ING ANTON SAVOV & DIPLOM MEDIA
SYSTEM DESIGNER ALEXANDER STEFAS

DDU — DIGITAL DESIGN UNIT
PROF. DR. ING. OLIVER TESSMANN
FACULTY OF ARCHITECTURE

RESEARCH & DEVELOPMENT PROGRAMME

Bau einer Laserschranke als Objektsensor for easy implementation in generic pick-and place robotic procedures

Hauptziel ist es einen güstigen plug-and-play Objektsensor für generische Greif- und Platzierroboter Prozeduren zu entwerfen. Ziel ist es, dass jeder nach dem Lesen dieser Arbeit diesen Sensor aufbauend auf einer Laserschranke nachbauen kann. Ein Möglicher Anwendungsbereich ist das 20.000Blocks (www.20000blocks.com) Projekt. In diesem Projekt baut ein UR10 Roboter von Minecraft-Spielern entworfene Gebäude als Modelle nach. Schwerpunkt dieser Arbeit ist die Kommunikation zwischen einem Arduino und einem Computer. Dabei stehen die elektrische Schaltung auf der Hardware-Ebene und die informatische Implementation auf der Software-Ebene im Mittelpunkt. Zudem wird eine Test-Simulation durchgeführt, die die Effektivität des Sensors zeigen soll.

PROBLEM STATEMENT

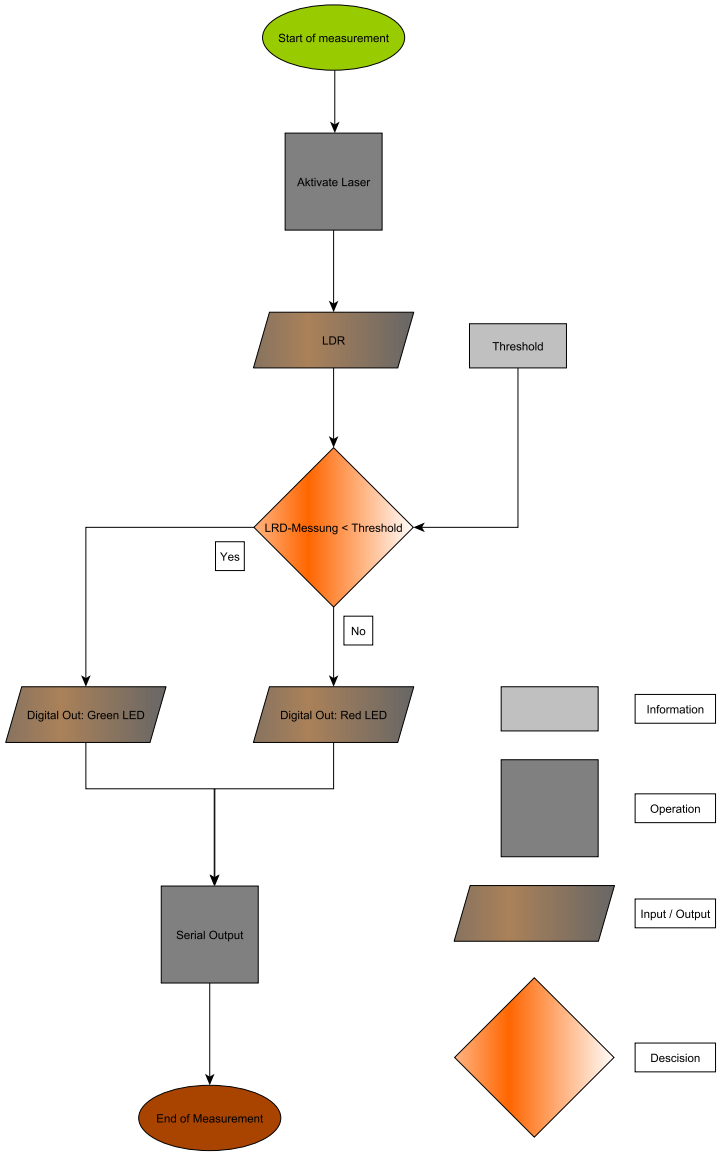
Die Nutzung von Robotern soll Menschen die Arbeit von repetitiven Abläufen ersparen. Roboter sind dazu ausgelegt, einmal programmierte Algorithmen beliebig oft auszuführen und immer bei gleichen Eingaben das Gleiche zurückzugeben. Hinzu kommt, dass Roboter beliebig komplexe Aufgaben lösen sollen, solange sie von den ihnen unterliegenden Algorithmen behandelbar sind. Diese, in der Informatik bekannt als “Korrektheit“, muss gegeben sein, dass ein Roboter Vorteile gegenüber Menschen bringen kann. Dabei zählen zu den Vorteilen, dass Maschinen oftmals präziser und gleichzeitig schneller sind als Menschen. Doch haben Menschen den meisten Robotern noch etwas vorraus. Sie können im Fall von Sonderfällen und unerwartet Problemen selbstständig reagieren, im Gegensatz zu den Robotern. Falls ein Punkt erreicht wird, der in der Implementation des Algorithmus nicht berücksichtigt wurde, ist es nicht mehr gegeben, das der Output des Prozesses noch einen sinnvollen Wert hat.

Das gleiche gilt auch für den Roboter Ginger, der 3D-Modelle im Rahmen des “20.000 Blocks” Projektes nachbaut. Im Projekt 20.000 Blocks werden durch das Spiel Minecraft Strukuren und Gebäude designt, die von Ginger aus kleinen Holzblöcken nachgebaut werden.

Nachdem ein Modell in Minecraft bereit zum Bau ist, wird das Objekt in Grasshopper, einem Plug-in für Rhinoceros, geladen. Darin wird das Objekt in einen Bauplan für den Roboter konvertiert. Vor dem Setzen jedes Blockes wird mit einem Unterdruck-Greifarm ein Block angesaugt und mit Hilfe von Klebepunkten im Modell verklebt. Die Blöcke werden am Ende einer Rutche entnommen, sodass immer ein Block griffbereit liegt. Ginger baut das Modell als einen Vollquader mit den maximalen ausmaßen des Objektes als Kantenlängen. Dabei werden jedoch nur ein Teil der Klötze miteinander verklebt. Die Restlichen Blöcke gelten nur als Unterstützung des Bauprozesses und werdem am Ende wieder entfernt werden, sodass das eigentliche Modell zum Vorschein kommt.

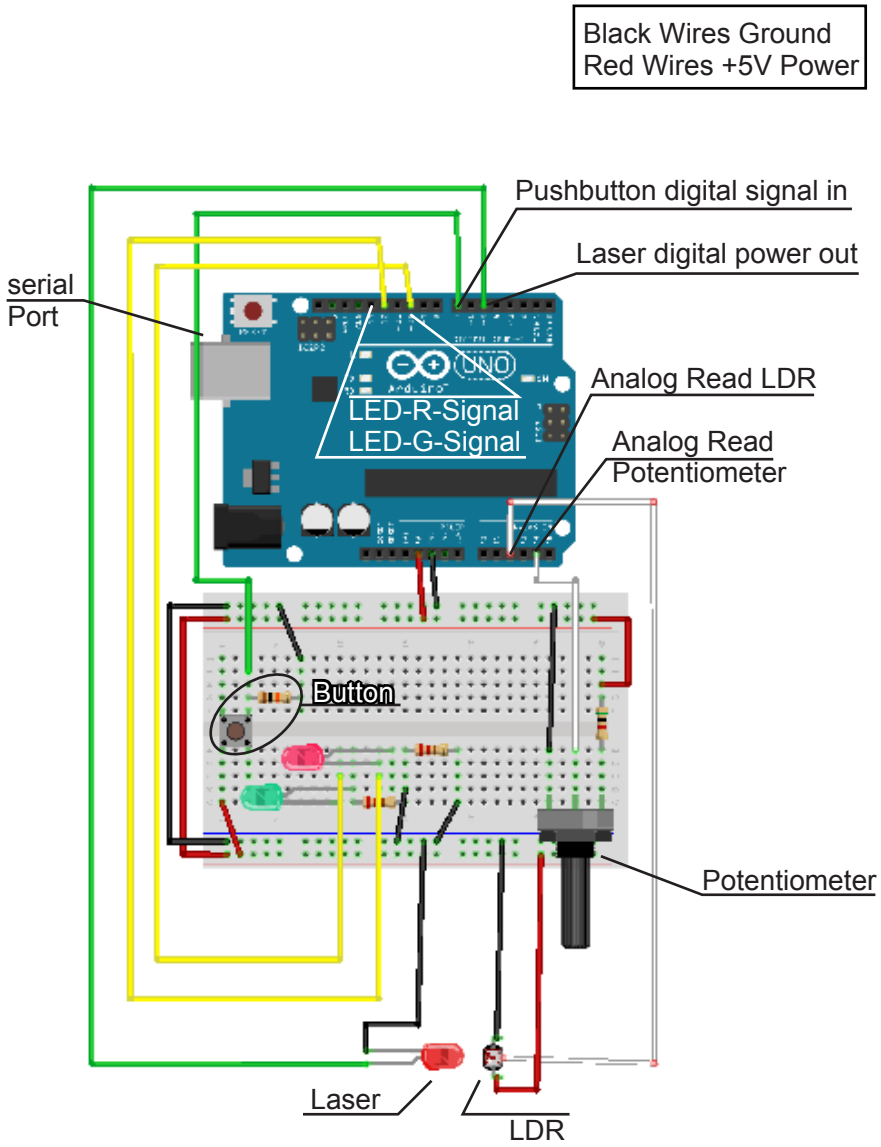
Nun gibt es die Möglichkeit, dass der Saugarm keinen Block ansaugt bzw. der Block nicht am Greifer hängen bleibt. Da der Roboter nicht erkennt, ob dies der Fall ist, ensteht durch das Fehlen eines Blockes einen Hohlraum im Modell. Blöcke die im Bauprozess eine Ebene darüber liegen,können dann u.U nicht mehr korrekt platziert werden, sodass das gesamte Modell unbrauchbar wird. Ohne passende Sensorik kann der Roboter diesen Fehler nicht erkennen und ein Mensch muss den gesamten Prozess überwachen. Das ist ineffizient, da es zum einen dem Prinzip des “korrekten” Prozesses widerpricht und den Menschen hindert, andere Aufgaben zu erledigen.

MESSUNG



Die Logik des Tests ist recht simpel. Der Vergleich des einteffenden Lichts gibt Aussage über Objekte, die sich zwischen LDR und Laser befinden. Nachdem die Laserdiode aktiviert wird, misst der LDR das aufreffende Licht am Auge und gibt es über einen analogen Pin an den Arduino weiter. Der gemessene Wert wird dann mit der Refernzmessung (im Bild: Threshold) bei ausgeschaltetetem Laser verglichen. Trifft mehr Licht auf das Auge Weist der LDR einen geringeren Wert auf als in Threshold hinterlegt. Wählt man den Threshold als aussagekräftige untere Schranke, kann man die Aussage machen, dass kein Objekt die Laserschranke blockiert. Dieses Testergebnis wird dann über serielle und digitale Känale nach außen kommuniziert.

ARDUINO SCHALTPLAN

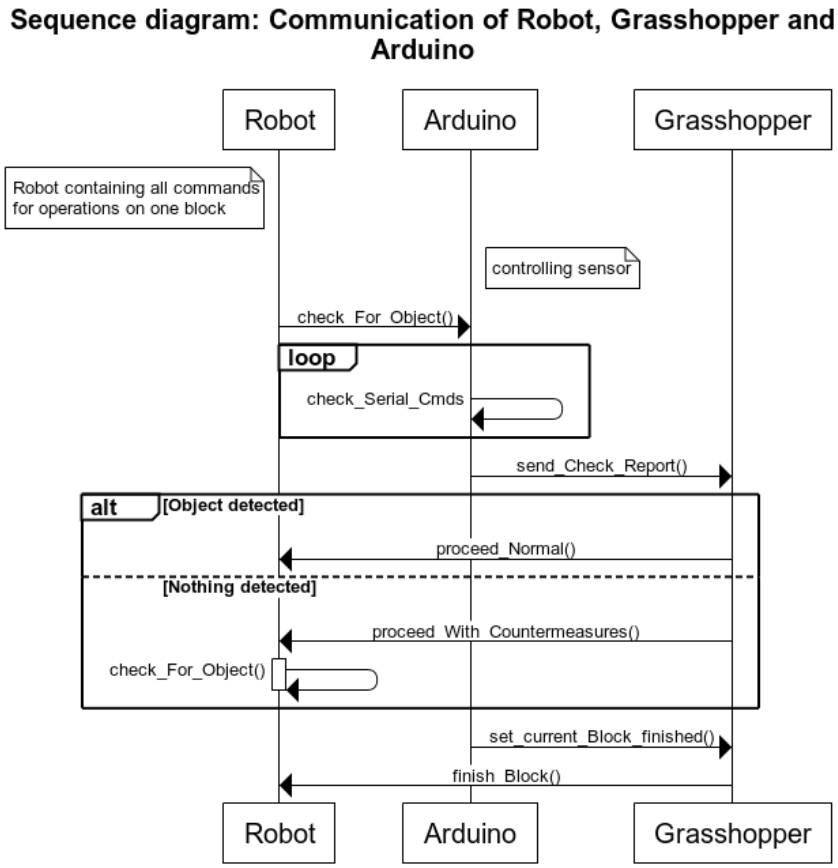


Das Auge des Roboters ist ein lichabhängiger Widerstand (light depenedant resistor - LDR). Das Licht liefert eine rote Laserdiode, die auf das Auge gerichtet ist. Per Signal - analog oder digital- wird eine Lichtmessung durchgeführt. Der Laser wird aktiviert und der LDR Messwert wird ausgelesen. Über das Potentiometer lässt sich ein Referenzwert des Lichteinfalls ohne Laser manuell ändern, der sonst automatisch beim Start kalibriert wird. Je nach Testergebnis wird der zugehörige digitale Ausgang (hier mit LEDs) angesteuert. Gleichzeitig wird über den Seriellen Ausgang ein Signal mit gleichen Informationen gesendet. Der Schalter kann dabei auch als Taster benutzt werden, welcher bei Aktivierung dauerhaft Messungen durchführt. Den Wechsel kann man neben der gesamten Bedienung des Senors während des Betriebs über serielle Kommandos durchführen. Die serielle Kommunikation ist jedoch nur mithilfe der Firmata von Alex Stefas benutzbar.
Für weitere Informationen: www. stefas@dg.tu-darmstadt.de

CODE

```
import SerialCommand

//variables
name for your pins
give a threshold

void setup(){
      //init your pins
      pushButton = input;
      LEDs, Laser = output;
      //setup your serial commands
      "NAME_OF_COMMAND" = callThisFunction();
      Start_Serial_Comunication();
}
void loop() {
      updateThreshold();

      IF Button_is_Pressed {
            Get  LDR_Pin_Reading();
            Send_Serial_Signal();
            IF LDR_Pin_Reading < threshold {
                  Light_Green_LED();
                  } ELSE{
                  Light_Red_LED();
                  }
            }

      }
      IF Button_is_Released {
      do_nothing();
      }
      Check_For_Serial_Command_Input();
      IF any {
      Process_Serial_Command();
      }
}
```

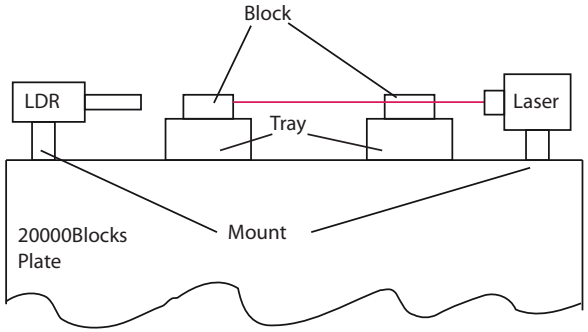
Das Gehirn bekommt der Roboter durch der auf dem Mirkokontroller ausgeführte Logik. Nach der einmaligen Ausführung der setup()-Methode wird die loop()-Methode wiederholt. In loop() wird erst nach einem neuen Schwellenwert gesucht, da dieser beim späteren vergleich benötigt wird. Danach wird abgefragt, ob der Button gedrückt wurde. Falls der Knopf gedrückt ist. Wird der Messwert des LDR ausgelesen und mit dem Schwellenwert verglichen. Da der LDR-Messwert kleiner wird, je mehr Licht auf ihn trifft, ist der Vegleich mit “kleiner”. Je nach Ausgang des Vergleichs wird eine LED angesteuert. Falls der Knopf zum Zeitpunkt der Abfrage nicht gedrückt wurde, wird nichts aufgerufen. Zuallerletzt wird im Input-Buffer nachschaut, ob Kommandos über den seriellen Port abgeschickt wurden. Diese werden dann gegebenenfalls noch ausgeführt bevor loop() erneut ausgeführt wird.



Der Kreislauf
Die Kommunikation des Sensors beginnt mit dem Roboter, der über digitales oder serielles Signal eine Messung anfordert, nachdem er versucht hat, einen Block zu packen und ihn zum Testort zu bewegen. Der Arduino sendet das Testergebniss gegebenfalls mit Evaluation an Grasshopper. Falls der Test positiv ausgefallen ist, gibt Grasshopper die Anweisung an den Roboter, normal fortzufahren, wie z.B den Block zu platzieren. Wenn jedoch kein Objekt erkannt wurde, werden von Grasshopper Gegenmaßnahmen an den Roboter gesendet, die von dem erneuten Versuch das Objekt zu greifen, über den Wechsel des Punktes zum packen von Blöcken, bis zum Stop der Anwendung reichen können. Falls ersteres gewählt wird, wird der Erkennungsprozess erneut durchgeführt, was im schlimmsten Fall in einer Endlosschleife enden kann. Falls der Roboter in endlicher Zeit ein Block gegriffen hat, kann der Prozess wieder normal fortgesetzt werden. Nachdem der Block fertig bearbeitet ist, wird der Abschluss des Prozesses vom Arduino Grasshopper gemeldet, damit Grasshopper dem Roboter die neuen Anweisungen für den folgenden Block geben kann. Damit beginnt auch die Teststruktur von neuem bis zum Ende des Gesamtprozesses.

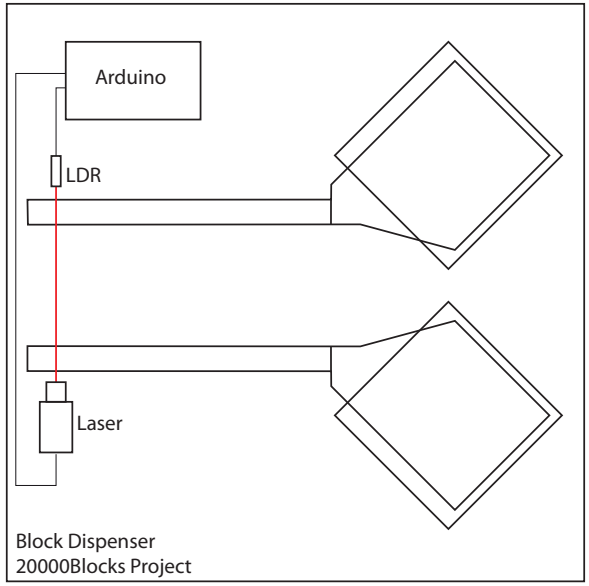
AUFBAU

Die Verwendung des Sensors ist beispielhaft für die Anwendung mit 20000 Blocks skizziert. Dabei werden die Blöcke in aus einem Vorratsbehälter auf eine Rutschbahn gesetzt. Die Blöcke rutschen an das Ende der Bahn, wo sie von Roboter aufgenommen werden. Ein möglicher Aufbau des Sensors ist rechts dargestellt. Der Sensor überprüft in dieser Aufstellung, ob der Block von der Rampe entfernt wurde und somit der nächste angefordert werden kann. Eine zweite Variante ist, den Sensor über die Blockrampe zu setzen. Damit kann der Sensor erkennen, ob der Block erfolgreich vom Roboter aufgegriffen wurde.



SETUP

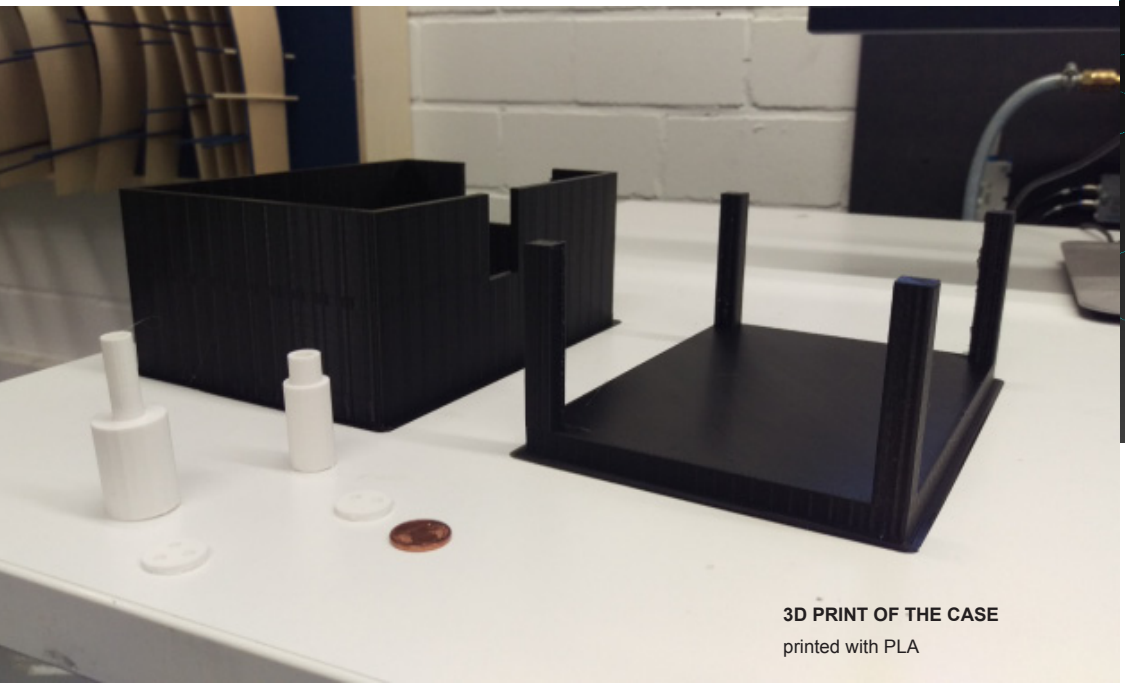
Front view



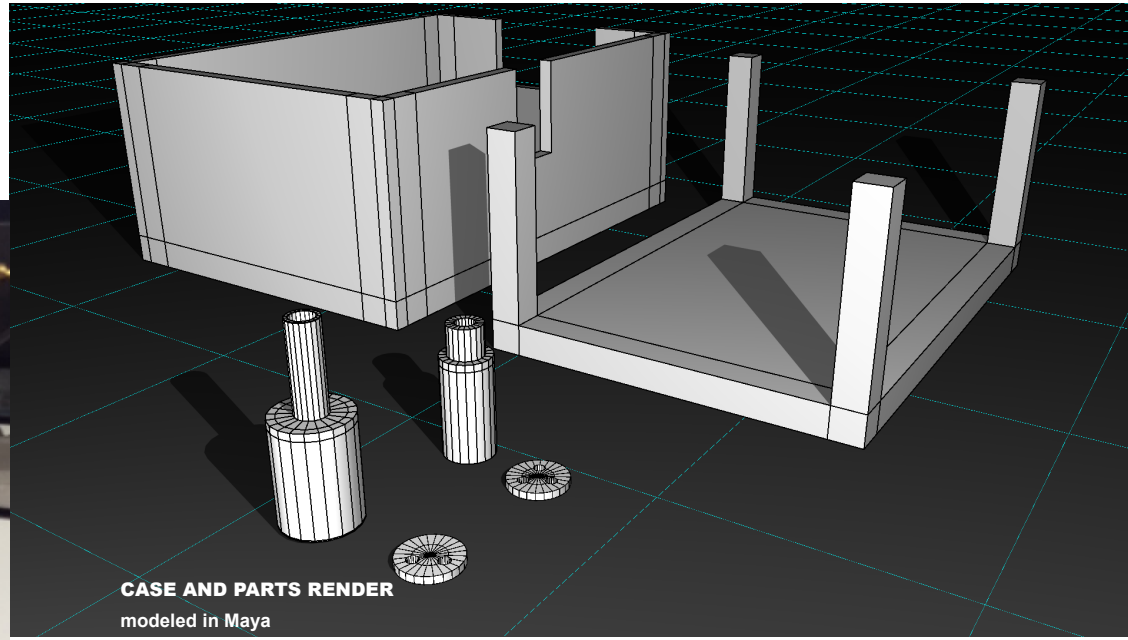
SETUP

Top view

GEHÄUSE



3D PRINT OF THE CASE
printed with PLA



CASE AND PARTS RENDER
modeled in Maya

ZUKUNFTSMÖGLICHKEITEN

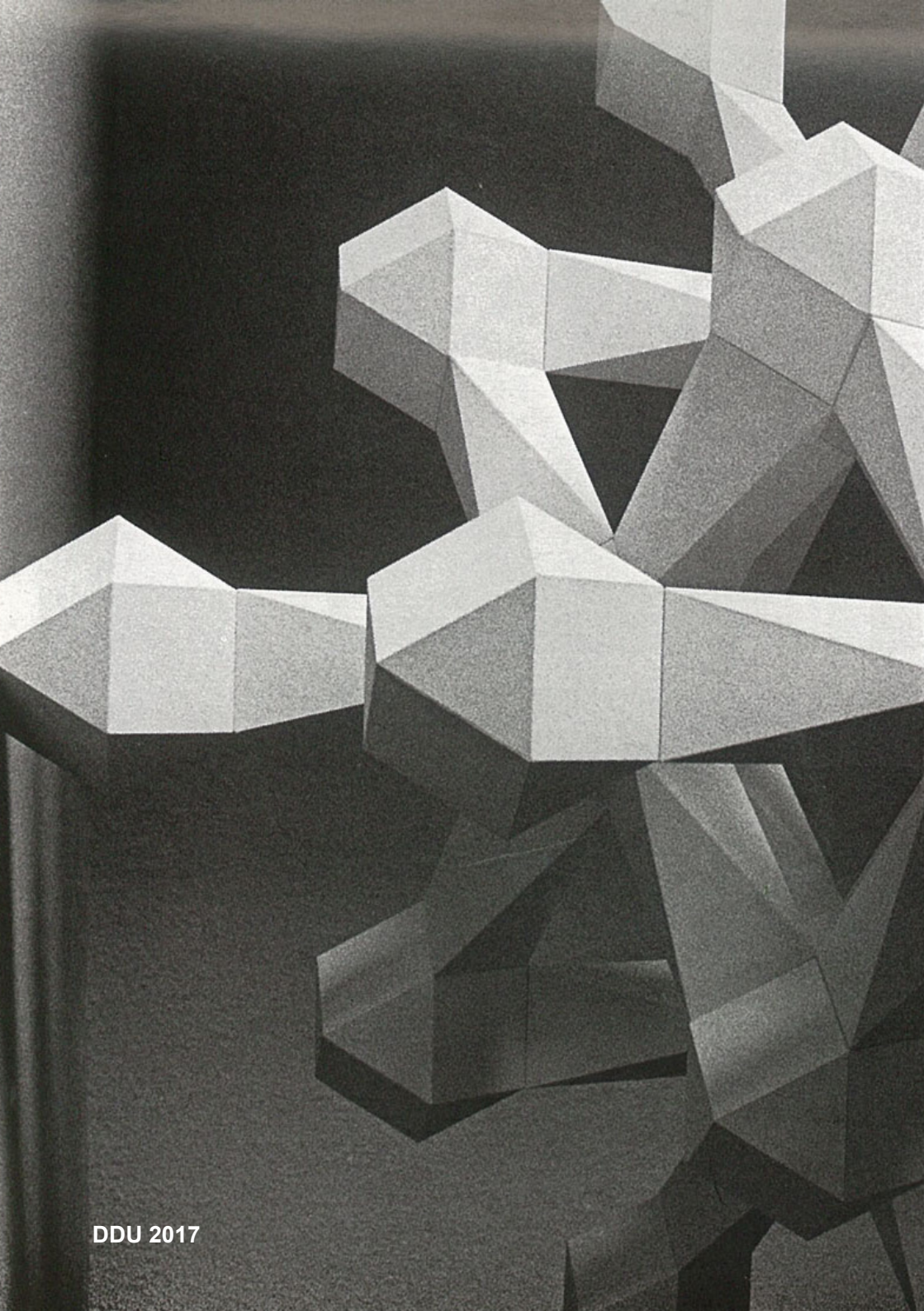
Erweiterungs- und Verbesserungsmöglichkeiten:

- Fehlersuche und Kalibrierung
  Einbau eines Displays auf dem Gehäuse, um Fehlersuche und Überwachung der Werte zu erleichtern.

- Erweiterung des Sensorapperates
  Steuerung von mehreren verschiedenen Sen-oren um den Aufbau des Arbeitsbereiches zu beschleunigen. Vorallem geometrische Kablibirierung des Roboterkopfes auf dem Baublock oder Zentrierung des Bauflächen-mittelpunkes.

- Einbindung in Bauprozesse
  Einbindung des Sensortests bei jedem zu ver-bauenden Block. Modulare Veränderung des Bauprozessen je nach Testergebnis.

- Einbau eines Fehlerprotokolls
  Überlegen eines festen Fehlerprotokolls, dass bei fehschlagendem Test abgearbeitet werden kann und der Roboter wieder in den geregelten Bauprozess überführt werden kann.


DDU 2017