Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

# Development of a Software for the Design of Electronic Circuits in 3D-Printable Objects

**Bachelor Thesis**
at the Research Group Technical Aspects of Multimodal Systems, TAMS
Department Informatics
MIN Faculty
University of Hamburg

submitted by
**Daniel Ahlers**
Informatics
6424701
on
10 Dec, 2015

| | |
|---|---|
| supervisors: | Dr. Norman Hendrich |
| | Florens Wasserfall |

# Abstract

3D printed objects with integrated electronics create new opportunities for the fabrication of devices. The lack of suitable design solutions is a problem that is not solved properly yet. This work compares the design processes of PCBs and 3D printed parts to combine these with the most important direct writing methods. In order to do that, the state of the art for 3D electronic design processes is analyzed and new approaches are developed. The most suitable approach is chosen by the requirements that are defined by the surrounding hard and software. The used hardware is a FFF 3D printer with a syringe based direct writing extruder. The chosen approach is implemented into the slicing software Slic3r. New file formats and data types are developed for the implementation. This work also includes a short manual for the developed software. The process and the software are evaluated and tested with some test circuits. All software in this work is developed as open source software.

# Zusammenfassung

3D gedruckte Objekte mit integrierter Elektronik ermöglichen immer neue Möglichkeiten bei der Fertigung von Bauteilen. Das Fehlen einer brauchbaren Design Software ist ein Problem, was bisher ungelöst bleibt. Diese Arbeit vergleicht den Design Prozess von Leiterplatten und 3D gedruckten Teilen, um es mit den wichtigsten Direct Writing Methoden zu vergleichen. Für diesen Vergleich wird der aktuelle Stand für 3D gedruckte Elektronik analysiert und neue Methoden entwickelt. Um die am besten nutzbare Methode auszuwählen, werden die Anforderungen an diesen Prozess definiert, welche hauptsächlich durch die benutzte Hard- und Software entstehen. Die benutzte Hardware besteht aus einem FFF 3D Drucker mit einem Spritzen-basiertem Extruder. Die ausgewählte Methode wird in die slicing Software Slic3r implementiert. Für diese Implementation werden neue Dateiformate und Datentypen entwickelt. Die Arbeit enthält außerdem eine kurze Gebrauchsanweisung für die entwickelte Software. Der Prozess und die Software wird mit einigen Test-Schaltungen analysiert und getestet. Die Software in dieser Arbeit ist als open-source Software entwickelt.

# Contents

# 1 Introduction

Since the invention of 3 Dimensional (3D) printing by [Hull, 1988] it has become more and more popular over the last years. This trend is mostly driven by cheap consumer 3D printers. With 3D printers it is possible to create objects that are impossible to produce with classic fabrication methods [Gibson et al., 2015]. 3D printing is used in medicine, in rapid prototyping and many other fields. It even makes the development of new products much more efficient and allows distributed manufacturing. [Palmer et al., 2004] showed the possibility to integrate wires into 3D objects, this method is called direct writing of conductive materials. In the following years many other approaches for direct writing were developed. The next step is 3D electronics, a combination of direct writing and 3D printing. With 3D electronics a product can be printed entirely instead of assembling the components in separate tasks. It allows to produce prototypes much faster and without the usual, expensive and wasteful Printed Circuit Board (PCB) production process. A benchmark circuit at NASAs Johnson Space Center showed that 3D electronics can reduce the volume of a circuit by 27% [MacDonald et al., 2014]. [Lopes et al., 2012], [MacDonald et al., 2014] and [Wasserfall, 2015] already mentioned that it is necessary to create software and algorithms for 3D electronic design. This unavailable software is a huge problem for the design process of 3D electronic devices.

## 1.1 Problem

The TAMS (Technical Aspects of Multimodal Systems) Group of the University of Hamburg has modified a 3D Fused Filament Fabrication (FFF) printer that is able to print 3D electronics [Wasserfall, 2015]. The printer prints conducting paths with a special silver paste directly into plastic parts. It can also place electronic components into the 3D object while printing. It uses a vacuum driven pick and place device to place these components. The pick and place component is supported by a camera to locate the components on a component tray. The placement of the components inside the 3D objects is done in a CAD software without any component reference or component footprint. Even the wiring had to be done by hand. The channels for the conductive paths and the spaces for the components in the 3D object have to be removed from the original object. This process is very complex and time consuming, therefore only a few simple test circuits have been developed so far. To make this process faster and simpler, a software to place electronics in 3D printable objects is needed for the operation of the 3D printer.

## 1.2 Objective of this Work

The objective of this work is to develop a software for the design of electronic circuits in 3D printable objects. To achieve that, the involved 3D printing process and the PCB design process should be analyzed. To find a suitable way to realize a new design process, the current approaches used by other researchers have to be analyzed. If there are no suitable approaches, new ones have to be developed. All the these approaches have to be evaluated to find the most suitable for the used FFF 3D printer. This approach has to be documented and implemented in a software. Wiring will not be implemented in this work.

## 1.3 Outline

Chapter 2 contains the basics for 3D electronics that are the 3D printing process, PCB design process and the most important direct writing methods.

In Chapter 3, the existing 3D design methods and some methods that are developed by the author of this thesis are shown. All approaches will also be evaluated in this chapter.

Chapter 4 contains the actual implementation of the software. It shows the requirements, the used method, the architecture, the used file formats and data types, and describes the actual implementation of the software. Lastly, it shows the integration of the software into the new design process and how the program can be used.

In Chapter 5, the benefits and the disadvantages of the software are shown, some tests are analyzed and the limitations of the used approach and the implemented software is evaluated.

The conclusion of the work and an outlook for future works is written in Chapter 6.

# 2 Basics for 3D Electronic Design

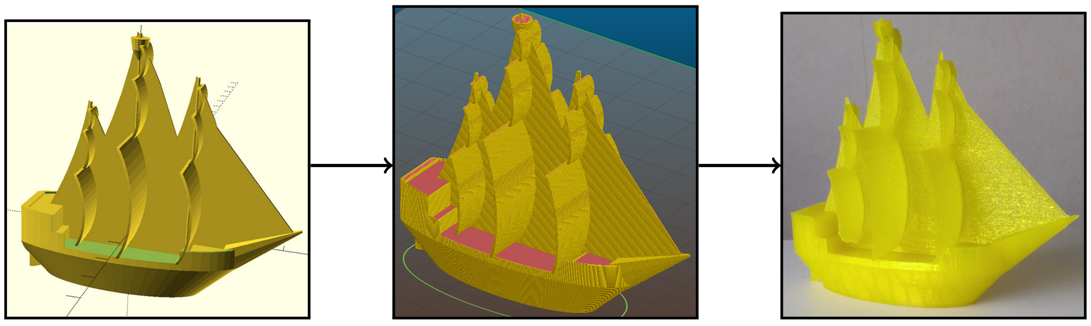
This chapter gives an overview of the 3D printing process, the PCB design process and shows the most important direct writing methods.

## 2.1 3D Printing Process

This section mostly describes the FFF 3D printing process. The printing process for other methods might be a little different. The process starts with the design of the model (figure 2.2(a)), then slices it (figure 2.2(b)) and prints the modeled object (figure 2.2(c)). The 3D printing process described in this section is based on the process described by [Gibson et al., 2015], [Horvath, 2014] and [Evans, 2012].

### 2.1.1 Model Design

The design process starts with the design of the 3D model. This model can be designed with a Computer Aided Design (CAD) software. It is also possible to generate the model with a laser or optical 3D scanner. To be printable, the model has to be a solid surface model; that means the model has to be a closed solid with no open edges. This is sometimes called watertight or waterproof model.



(a) Model design (b) Slicing (c) Printing

Figure 2.1: The 3D printing process of a ship designed with OpenSCAD and sliced with Slic3r.

For the 3D printing process, the STL file format is widely used. STL stands for STereoLithography and is developed by 3D-Systems in 1988. The STL format is a triangle based format. That means the surface of an object will be completely represented by triangles. Each triangle is described with its own three points. This leads to a lot of redundant information. Nearly every CAD software can export models in the STL format. The STL format is the basic for the calculation of the slicing process presented in the next section.

### 2.1.2 Slicing

With the STL file, created in the model design, the object can be sliced. The slicing process is necessary because printers cannot process 3D models on their own. To make the 3D object printable for the printer, the slicer cuts the model into horizontal slices. The hight of these slices is defined by the layer thickness. Then the slicer generates a path for each layer which the printer follows later. This path is called toolpath and it is described by a number of simple instructions so the printer knows where to move and when to add material. These instructions are called g-code. The slicer also adds instructions for the temperature and the flow rate of the printing material to the g-code. The slicing software holds most of the configuration information for the 3D printing process for example the print speed, the quality and the resolution. Features like infill for the object or support structure for overhang structures are also generated by the slicer.

### 2.1.3 Printing

With the generated g-code from the slicer, the printer can start the printing process. The g-code will be executed step by step by the printer. The printer is controlled by the control software and the printer firmware. Most cheap 3D printers work with the principle of Fused Filament Fabrication (FFF). These printers work with a cartesian robot driven by stepper motors. The printer can be moved along the x-, y- and z-axis so that the extruder can access every point in the workspace. The extruder is the printhead of the printer. It dispenses the material as it is instructed by the g-code from the slicer. Most FFF printers use plastic from a plastic spool that is melted and forced through a printing nozzle. The printer prints on a print bed. The print bed is a flat surface that is sometimes heated to prevent warping and cracking while cooling down and providing better adhesion of the first layer. When the printer has correctly done all instructions form the g-code, the 3D printed part is finished.

### 2.1.4 3D Printing Toolchain

Each step of the 3D printing process is done by a different tool. The main tools are CAD software, slicer, control software, firmware and electronics. There are some combinations of these basic tools available, but they do the same. The design in section 2.1.1 is done with a CAD software like OpenSCAD or Autodesk Inventor. The slicer does the slicing described in section 2.1.2. Popular slicers are Cura, Slic3r and KISSlicer. Control software, firmware and electronics are described in section 2.1.3. The control software sends g-code commands to the printer firmware, and the printer firmware executes the command with the electronics..

## 2.2 PCB Design Process

The first Printed Circuit Board (PCB) was produced in 1942 and since then they have become very important. Today PCBs are used in nearly every electronic component. PCBs provides the electrical connections and the mountings for the attached components. Goal of the PCB design is to produce a cheap circuit that fulfills all given electronic requirements. The process starts with the specification, followed by the schematic design (figure 2.2(a)). Afterwards comes the placement (figure 2.2(b)), the routing (figure 2.2(c)) and it ends with the production of the PCB (figure 2.2(d)). The PCB design process in this section is based on the process described by [Jansen, 2001] and [Ammon, 1987].

### 2.2.1 Specification

The design process of a PCB starts with a specification of the board. It determines what the design should achieve and not how this could be done. A specification should contain at least the customer needs, electrical requirements and physical parameters. Customer needs are, for example, the behavior of the designed circuit and constraints for the design. Electrical parameters are, for example, the signals that are input and output, the power source and how much energy the PCB should consume. Physical parameters are, for example, the size and the weight of the PCB and even environment parameters like the temperature and the humidity.

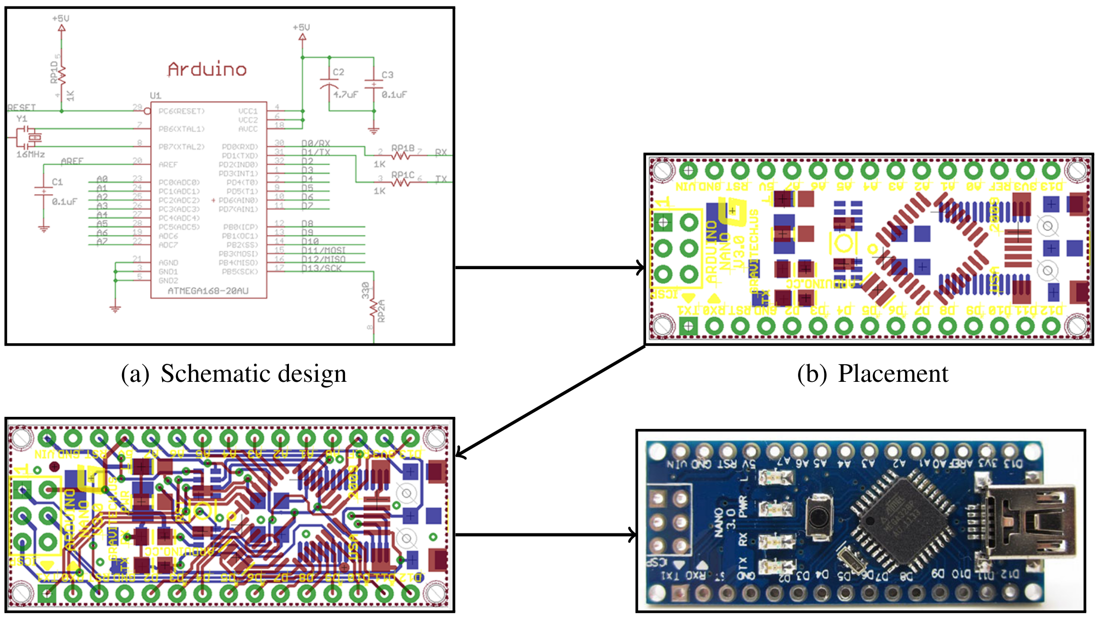

(a) Schematic design
(b) Placement
(c) Routing
(d) Production

Figure 2.2: PCB Design Process of an Arduino Nano designed with Eagle.

### 2.2.2 Schematic Design

When the specification is finished, the process continues with the schematic design. This design is done by creating a circuit diagram. This diagram contains the electronic components and their connections. Components are selected out of a component library which contains a description of the electronic properties of the component. The placement in the schematic has no reference to the placement on the PCB. This schematic is the basis for some simulations and electrical rule checks. Simulations are used to verify the specifications and the logical and timing behavior. Electrical rule checks are, for example, checks for unwired pins or shorts. The finished schematic design is saved as a netlist for further design. The netlist contains components and their interconnectivity information.

### 2.2.3 Placement

With the finished schematic design, the components can be placed on the PCB. Each component of the schematic design has to be placed on the surface of the PCB board. The placement can be done manually or automatically. Since the automatic placement often does not fit all requirements, the placement can be done semi-automatically by editing the automatic placement or placing the main components and use the automatic for the rest. The PCB design software controls the compliance of the given design rules. Design rules are, for example, the minimum distance between components. The placement is critical for the routing, therefore, it is very complex and time consuming. To visualize the connections of the components, rubber banding is used. Rubber bands are straight lines between pins of the component that represent the needed connections. When all components are placed, the routing can start. It is possible that the placement has to be changed during the routing process because a bad placement leads to a very long and complex routing.

### 2.2.4 Routing

Routing is the actual connection of the placed components. It is also called wiring even if there are no real wires involved. The main goal for the routing is to find a short net that connects all components and meets all requirements. Wires can have different widths and can be placed on different layers of the PCB. These layers are independent and just connected at pins or vias. Vias are connections between layers. Wires can not cross each other therefore, it is very complex to place the wires properly. The routing can be done manually or automatically by an autorouter. Autorouters are algorithms that automatically connect all components with wires. They are NP-hard so there is no best result, therefore, most autorouters use heuristic algorithms. The PCB design software provides design rule checks like distance between wires and many more. The software can also check if all connections of the schematic design are connected properly. With a correct placement and routing, the design of the PCB is done.

### 2.2.5 Production

For the production, the design software provides some production data. Examples of this are the layout of the different layers, a solder resist mask or a drilling plan. The production of the PCB is mostly done by the following procedures:

- Silk screen printing: Printing the layout with etch-proof ink and remove the uncovered areas with chemical etching.
- Photoengraving: Using UV sensitive PCB, transfer the mask and remove uncovered areas with chemical etching.
- PCB milling: Remove the unnecessary areas with a CNC mill.

When the PCB board is finished, the components can be soldered to it by hand or with a placement and soldering machine. It should be tested if the finished board fulfills the specifications.

### 2.2.6 PCB Design Toolchain

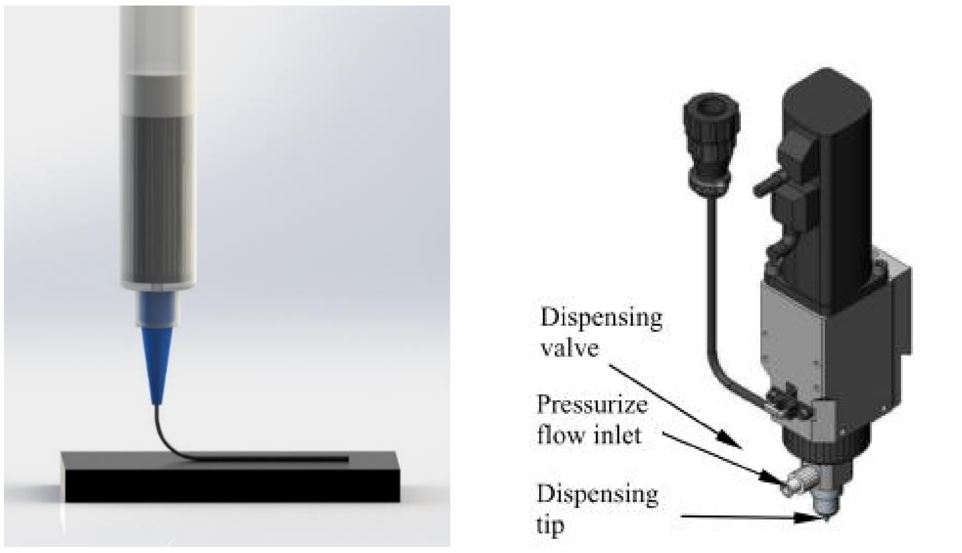
Most PCB design tools are covering both: schematic design and layout. Popular tools are: Eagle, Altium Designer or KiCad. Every tool can export Gerber files to produce the PCB. Gerber files just contain the masks of the layers without any reference to the electronic components.

## 2.3 Direct Writing

To enable electronic circuits in 3D printable objects, wires for the electronics are necessary. These wires can be placed with direct writing of conductive materials. The integration of these wires into the material of the object is possible because of the layer by layer process of 3D printing. Direct writing means selectively placing conductive materials directly onto the underling substrate to generate footprint and wires for embedded electronics. This is the key factor to replace the traditional PCB production methods. There are three main methods to place wires with direct writing. All methods place conductive wires in a different way. The methods are fluid extrusion through nozzle, inkjet based droplet ejection onto the substrate, and jetting aerosolized material with a focused stream onto the substrate.

### 2.3.1 Extrusion Based Methods

The extrusion based method uses pressure to dispense the material in semifluid form through a nozzle onto the surface. This is either possible through a syringe (figure 2.3(a)) driven by a stepper motor or a micro-dispensing pump (figure 2.3(b)). The key factor to place a wire properly is a dynamic level of pressure during the operation of the printer. A precise pressure is necessary to place the wires in a defined width. Like in the 3D printing process, a vacuum for the retraction of the material at the end of the trace improves the extrusion line finish. The extrusion based method is typically used with semifluid inks that are filled with small conductive particles. These are often silver or gold particles due to their conduciveness and their oxidation resistance. Carbon based inks are also used but they are less conductive. The more material is loaded into the ink the more conductive it is. Yet with more material, the ink gets less fluid and is



(a) Syringe based extrusion (b) Micro-dispensing pump extrusion

Figure 2.3: Schematic of extrusion based dispersion [Perez and Williams, 2013]

harder to process. The extrusion based method is relatively easy to integrate into the printer, especially the low tech syringe solution. This based method is, for example, used by [Medina et al., 2005] and also at the University of Hamburg [Wasserfall, 2015].

### 2.3.2 Inkjet Based Methods

The inkjet based method is very similar to inkjet based desktop printers. It drops material onto the substrate where it is needed. For the dispersion either a piezoelectric or a bubble jet dispensing method can be used. A piezoelectric dispenser works with an expanding diaphragm to dispense a single drop of ink, as it is shown in figure 2.4. The bubble jet dispenser heats up the ink to generate a drop of ink due to the expansion of the ink. Inkjet based methods also use metal or carbon filled ink that is more fluid. Because of that, it has much lower particle density which leads to lower conduciveness of the printed wire. The inkjet method is capable of much higher resolution than the extrusion based method and can print in different angles. The method is further described by [Walker and Lewis, 2012].

### 2.3.3 Aerosol Jet Based Methods

The aerosol jet based method uses a slightly different approach. The conductive metal or carbon fluid is aerosolized into a vapor stream. The vapor is accelerated through the

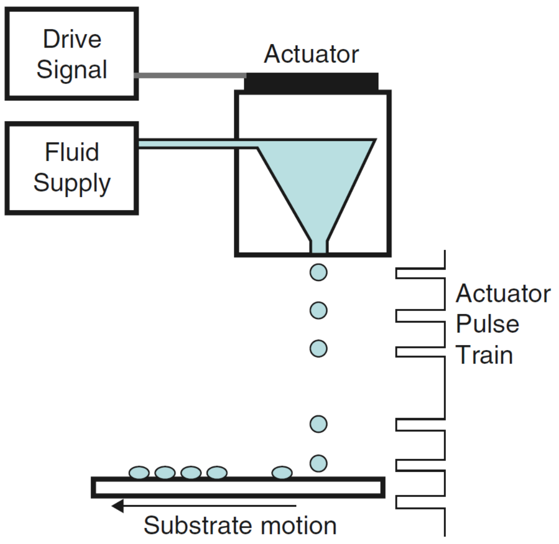


Figure 2.4: Schematic of inkjet dispersion [Gibson et al., 2015]

nozzle and hits the surface. When the aerosol hits the surface, the metal sticks there. The fluidity is not an issue with this method as long as the vaporization is possible. The sticking layers are very thin. To get thicker layers, the process has to be repeated several times. This makes the deposition rate very slow. This method is further described by [Goth et al., 2011].

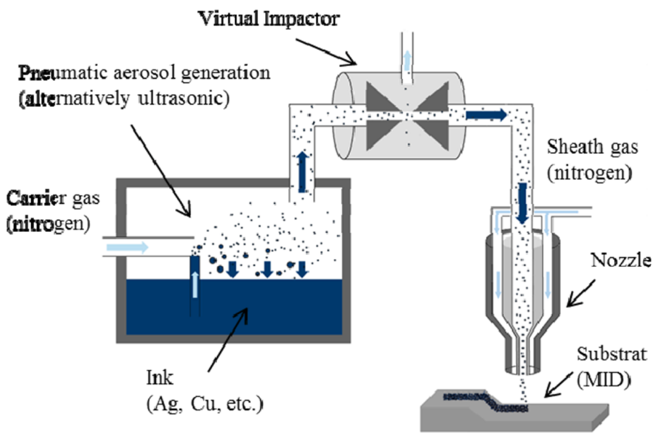


Figure 2.5: Schematic of aerosol based process [Goth et al., 2011]

# 3 Methods for 3D Electronic Design

This chapter contains an overview of the currently used approaches to design electronic circuits in 3D printable objects. It also shows some new approaches developed by the author. Each approach is described shortly and evaluated afterwards. The final selection of the most suitable approach will happen in section 4.2.

## 3.1 State of the Art

The design of 3D printable electronic circuits has been a problem since the first direct writing circuits from [Palmer et al., 2004]. Since then many researchers have solved this problem in different ways which will be described in this section.

### CAD Design

The CAD design approach uses a 3D CAD software like SolidWorks, OpenSCAD or Autodesk Inventor to design the whole circuit. This simple approach is used by [Medina et al., 2005], [Lopes et al., 2006], [Navarrete et al., 2007], [Lopes et al., 2012], [Shemelya et al., 2015] and for first prototypes at the University of Hamburg [Wasserfall, 2015]. All components like parts, wires, footprints are designed like this. The final model will be saved into distinct objects, the circuit object and the model object, and saved as STL files for the printing process.

Using only CAD software to design the whole device is a good method in order to test the printer's hardware. It takes a lot of effort to develop an extra software which should be done after the development of the printer is completed. Exporting the design as an STL file is also possible without modifications to the software. Even real 3D circuits and rotations around all three axis are possible. All tools of the toolchain and the printing method are exchangeable. Designing a circuit this way is very complex and nearly unusable. It is also very time consuming. There is no reference to electronic parts and there is also no part library.

## PCB Design with CAD Support

PCB design with CAD support uses a classical PCB design software to design the circuit on a predefined two dimensional workspace. Then the circuit is saved and modified in different ways.

[MacDonald et al., 2014] presented a method where the circuit is designed as an unfolded outer surface and then deformed by hand with a 3D CAD software. It was folded around the six sides of a cube or wrapped on the surface of a cylinder. This approach does not allow interconnections between different layers due to its 2D limitations. With this approach the electronic circuits must lie on the surface of the object.

[Espalin et al., 2014] also deformed the 2D PCB circuit design with a traditional 3D CAD software. The finished electronic design is saved in a STL file as an input file for the 3D printing process.

[Gutierrez et al., 2011] designed multiple flat circuits with a 2D PCB design software and saved them in a format that is readable with a 3D CAD software. The electrical interconnections are captured in a distinct file. The files were edited with a 3D CAD software and adjusted to fit the shape of the 3D object and stay connected to each other through the interconnect file. The 3D CAD software gives the user full control over the circuit's positions. When the design is finished, the circuit is saved as an STL file for the printing process.

[Swensen et al., 2015] used the PCB design software Eagle for the design of their circuits. The PCB layout is computed with a MathLab program to generate STL files for the printing process. The MathLab program also generates inlets and outlets for their direct writing method.

Circuits designed by a classic PCB design software can profit from the well tested PCB design software and its possibilities to test and simulate circuits. There is a reference to the parts and an existing part library. With this process the PCB design software, the 3D printing tools and the printing methods are completely exchangeable. It is impossible to print real 3D circuits with this method and only possible to print 2D circuits or pseudo 3D circuits. It is also not possible to rotate single components around all 3 axis, it is just possible to rotate the complete PCB design. The process is a bit faster than the CAD design approach but it is still complex and not very useful.

### Autodesk Project Wire

Autodesk Project Wire is an announced commercial design tool for 3D printed electronics [Autodesk, 2015]. The software is closed source and not available yet. It is a browser application that will be integrated into the Autodesk Spark platform. The Spark platform is a cloud based 3D printing platform and is also not available yet. Autodesk Project Wire is a full 3D electronic design software where components can be placed freely in 3D space and be rotated around all 3 axis. It contains a library of components and has a tool for 3D wiring. The components and circuits are automatically subtracted from the imported 3D model. Wires can also be placed on the surface of the model for capacitive sensing. To use Autodesk Project Wire and its printing function, the full Autodesk toolchain has to be used.

Autodesk Project Wire provides full support of placing electronic components in 3D space. It has a direct reference to the parts and a part library. Circuits that are created are real 3D circuits, with the support of rotation around all 3 axis. Wiring the components in 3D space is also possible. It is even possible to print 3D components like capacitive sensors. The slicer can slice circuits to print them with printers that support direct writing. To achieve that, the full Autodesk toolchain has to be used. Autodesk Project Wire is fully commercial and not released yet. It is also a browser integrated tool, therefore all work has to be done in the cloud and the user has to trust Autodesk with his data.

## 3.2 Possible new Approaches

Since manual CAD manipulation is complex and not useful and Autodesk Project Wire is a commercial software which is not released yet, some new approaches may be suitable.

### Multilayer PCB Design

Multilayer PCB design uses a PCB design software with multiple layers. Each layer from the design software represents a slice in the 3D object. The multilayer design is saved as a PCB design. To print this circuit, a software is needed that converts the PCB design into a multilevel circuit. Every layer of the PCB design has to be on a different slice in the 3D object. For interconnections between the layers, the vias of the PCB design are used. The output of this software has to be readable by the slicing software.

With Multilayer PCB design it is possible to print stacked 2D circuits with full support of electronic components out of part libraries. Even the wiring and all simulations can be done in the PCB design software. Since all PCB design software supports standard exchange formats for PCB production, the PCB design software is exchangeable. The 3D printing tools and the printing methods are also exchangeable. The direct reference to the parts of the design are not present in the 3D printing tool. The number of layers has to be configured manually and are limited in most PCB design products. It is not possible to rotate the components around all 3 axis and it is also not possible to print real 3D circuits because the design is just stacked 2D circuit connected together. 3D components like capacitive sensors are also not usable with this approach.

## PCB Design Software with 3D Support

PCB design software with 3D support is an open source PCB design software with an added 3D electronic design feature. To achieve that, the whole design process of the PCB design software has to be changed. For a better reference it should be possible to import the 3D model. All components have to be designed in 3D and all formats have to be suitable to handle the 3D information. All algorithms, like the autorouter, have to be deactivated or modified to handle 3D electronic design. The output of this software has to be readable by the slicing software.

A PCB design software with integrated 3D placement and 3D wiring can create real 3D circuits. Direct reference between the placed component and the part of the library is possible. It can also integrate 3D components like capacitive sensors. Some of the simulations of the PCB design software can still be used. The 3D printing tools and printing methods are still exchangeable. The PCB design software is not made for 3D design, so it is complex to develop this approach. Most PCB design software is not open source, so it can not be modified. Many features like the autorouters are unusable because they are made for 2D designs. Since the PCB design software is changed intensively, it can not be exchanged. It is also not possible to get any automatic reference for the structure and thickness of the layers or the actual printing path from the slicing software.

### Independent 3D Electronic Design

Independent 3D electronic design is a completely new developed 3D electronic design software. The circuit schematic can be done in a PCB design software and can then be imported into the new software. The new software has to place the components defined in the schematic on different levels and rotate around 3 axis. For a better reference it should be possible to import the 3D model. Wiring of the connections, defined in the schematic, has to be possible in 3D space. The output of this software has to be readable by the slicing software.

An independent 3D electronic design software can create real 3D circuits with rotations around all 3 axis. The simulations for the schematic in the PCB design software are still usable and the PCB design software is exchangeable. The 3D printing tools and printing methods are exchangeable. Even 3D components like capacitive sensors are possible. This concept has no features from the PCB design software and has to be developed from scratch. This development is very complex and time consuming. It is not possible to get any information about the slice structure from the slicing software.

### 3D Electronic Design in Slicing Software

3D electronic design in slicing software is an option where the 3D electronic design is integrated into an open source slicing software. To do that, the circuit schematic is created in a PCB design software and then imported into the 3D design software. The slicing software has to be modified so that it can import these schematics. The components and connections defined in the schematic can be placed in 3D space. In order to print the designed circuit, the design has to be converted into a format that is readable by the slicer or the slicer has to be modified so that it is possible to process the design with the slicer.

The integration of 3D electronic design into the slicing software can create real 3D circuits with rotations around all 3 axis. Simulations of the schematic in the PCB design software are still usable. Components and wires can be aligned to the printing path. The slicing software can even react to the footprint and wires and can treat them differently in the printing process. For example, it is possible to create channels for wires to prevent shorts. It is possible to print 3D components like capacitive sensors. The PCB design software is exchangeable with little effort. The whole implementation is not very time-consuming due to the existing 3D representation for components and the interface of the slicer. The modified slicer is obviously not exchangeable and it supports just the printing methods that the used slicing software supports. The used components have to be imported from the PCB design software to the slicing software. This method can not profit from features provided by the PCB design software like autorouters.

# 4 Implementation

In this chapter, one of the approaches of the previous chapter will be implemented. To choose one method and the surrounding toolchain, the requirements are listed. After that, the architecture of the software is defined. Then the file formats and data types are specified. With all of this the implementation can be done. At last, the integration into the new design process and the program usage will be shown.

## 4.1 Requirements

The requirements for the implementation can be grouped into three groups. The functional requirements, nonfunctional requirements, and the constraints. Functional requirements describe interactions between the system and its environment. Nonfunctional requirements describe aspects that are not directly related to functional behavior. Constraints are defined by the environment of the software. The requirements for the software to design electronic circuits in 3D-printable objects are:

- Functional requirements:
  - The object should be displayed as a reference to place the components.
  - It should be possible to place a component in the 3D space of the object.
  - The user should be able to set the position of the component precisely by coordinates.
  - To align the component, the planned toolpath of the object should be displayed.
  - It should be possible to rotate the component around all three axis.
  - Saving and loading the design should be possible at all times.
  - The layer height of the footprint should be the same as the height of the layer on which the component is placed.
  - The height of the footprint should be customizable.

  - The component should be displayed as a different 3D volume than the components footprint.
  - The size of the component should be customizable.
  - It should be possible to remove the components later in the design process.
  - The modification of the given component should be possible.

  The functional requirements are mostly driven by the usability of the actual design process. Some are due to the limitations of the used direct writing method or the used FFF 3D printer.

- Nonfunctional requirements:
  - The object should not be modified during the design process.
  - The implementation should be expendable to different input types.
  - The components should be chosen from a library.
  - It should be possible for the slicer, to treat the components different than the other parts of the printed object.
  - The 3D representation should be fast with a low latency.
  - The chosen software should be upgradeable.
  - The design process should be fast and easy.

  Most of the nonfunctional requirements are due to the usability, upgradeability and extensibility. Some, like the different treatment of the slicer, are due to the printing process.

- Constraints:
  - The implementation should be open source.
  - The implementation should work on different platforms.
  - The implementation should work with the used FFF-Printer.
  - The implementation should work with the used extrusion based direct writing method.
  - The implementation should use Slic3r as a slicing software.

  The constraints are driven by the open source culture of the whole process and the already existing surrounding hardware.

## 4.2 Selected Method and Software

The **CAD Design** and the **PCB Design with CAD Support** methods are unusable due to their high design effort. **Autodesk Project Wire** is still unavailable and uses their own closed source infrastructure. A **Multilayer PCB Design** would not be able to produce real 3D circuits and is also limited by the PCB design software. The missing 3D representation in the method **PCB Design Software with 3D Support** leads to a very high development effort. The **Independent 3D Electronic Design** method has also no 3D representation so the development effort is also very high. The method **3D Electronic Design in Slicing Software** is a suitable way to solve the problem. Although this method does not support all printer types in one software, this method is a suitable solution for the used FFF printer. It also has a relatively low development effort due to the existing 3D representation in the software. With this method it is also possible to modify the slicer so it can treat the component's footprint and wires different than the rest of the object. Even all the other requirement defined in section 4.1 can be achieved with this method. Therefore, the method **3D Electronic Design in Slicing Software** will be implemented in this work. To do that, the surrounding software has to be chosen.

### CAD Software

The CAD software is fully exchangeable with this method. So design of the 3D object can be done in any CAD software with STL file support.

### PCB Design Software

The PCB design software should be exchangeable too but there is no standard file format for the exchange of PCB schematic designs. The implemented software should be capable of handling multiple file formats. But, for the first implementation, a first format has to be chosen.
KiCad is an open source PCB design software that can be used on multiple platforms. It has a modular layout but is not widely used. The saved PCB schematic designs does not contain the necessary footprint information. This information is stored in the component library. This library is spread in different files and in multiple locations. Some components are even loaded directly form GitHub when they are needed. Due to this complex component library system this method is not suitable for the first implementation.
Eagle is a proprietary PCB design software. It is widely used and is available on multiple platforms. All the information of the PCB schematic design is stored in an XML

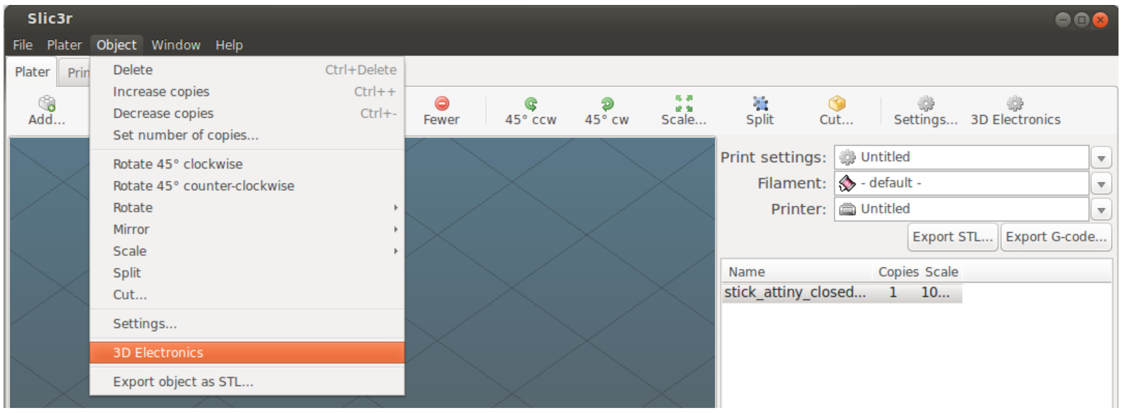
file. This file contains the needed libraries and all necessary information about the components and their connections. Although Eagle is not open source, it is the best format for a first implementation.

### Slicing Software

The slicing software has to be an open source software because their source code is available and the license allows modifications. There are plenty of open source slicing software products like Slic3r, Cura and many more. Slic3r is a slicing interface with an integrated slicing engine. It is written in Perl with some core algorithms and data structures written in C. It already has the needed 3D representation and is widely used. Due to this and the requirement to use Slic3r it will be used here.

## 4.3 Architecture

The possibility to place 3D electronics in 3D printable objects will be integrated in the slicing software called Slic3r. To start the editing of the 3D electronics, a new button and a menu entry will be integrated in the Slic3r interface. The button and menu entry are shown in figure 4.1. The button will open the 3D electronics editor for a selected



Print settings: Untitled
Filament: - default -
Printer: Untitled

| Name | Copies | Scale |
|---|---|---|
| stick_attiny_closed... | 1 | 10... |

Figure 4.1: The “3D Electronics” button on the top right side and the “3D Electronics” menu entry in the “Object” menu.

object. The editing will happen in an extra window specially designed for this purpose. The class structure of the extra window is shown in figure 4.2. The extra window will be called by the main window in `Slic3r::GUI::Plater`. Classes that already exist in Slic3r are colored gray. New classes for the integration of 3D electronics are black. They are grouped into Graphical User Interface (GUI), file handlers, data types, and

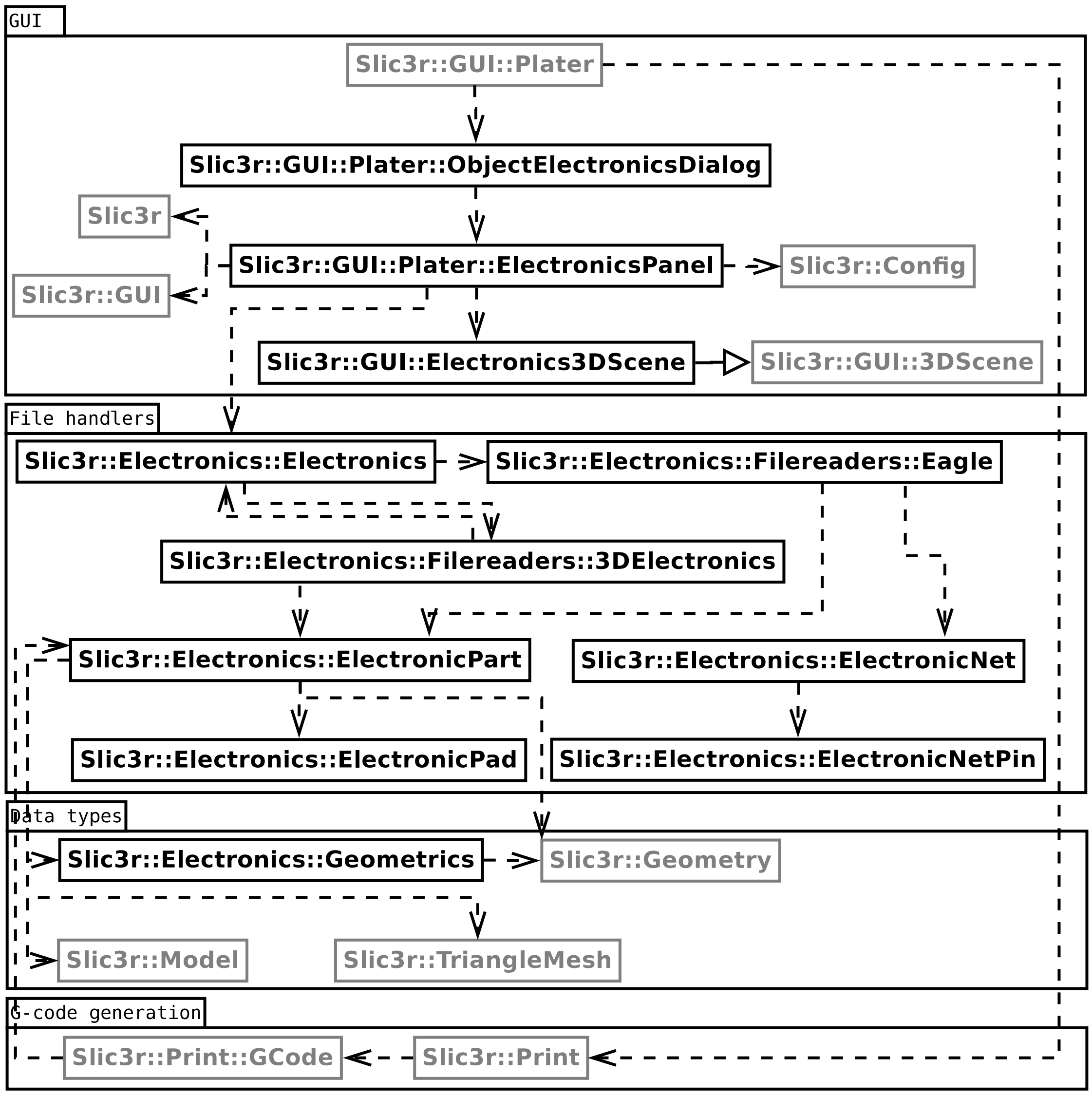

GUI
Slic3r::GUI::Plater
Slic3r::GUI::Plater::ObjectElectronicsDialog
Slic3r
Slic3r::GUI::Plater::ElectronicsPanel
Slic3r::Config
Slic3r::GUI
Slic3r::GUI::Electronics3DScene
Slic3r::GUI::3DScene
File handlers
Slic3r::Electronics::Electronics
Slic3r::Electronics::Filereaders::Eagle
Slic3r::Electronics::Filereaders::3DElectronics
Slic3r::Electronics::ElectronicPart
Slic3r::Electronics::ElectronicNet
Slic3r::Electronics::ElectronicPad
Slic3r::Electronics::ElectronicNetPin
Data types
Slic3r::Electronics::Geometrics
Slic3r::Geometry
Slic3r::Model
Slic3r::TriangleMesh
G-code generation
Slic3r::Print::GCode
Slic3r::Print

Figure 4.2: Class structure of the software. Grey classes already exist in the Slic3r software, black classes are new for the integration of 3D electronics. The classes are grouped into Graphical User Interface (GUI), file handlers, data types, and g-code generation.

g-code generation. The GUI classes mainly manage the GUI and the interaction with the user. The file handlers read and write the different file formats that are shown in the next section. The data type classes provide the data types and some specific methods and are presented in section 4.5. The g-code generation classes generate the g-code for the printer when the modeling is done.

Listing 4.1: Eagle format

```
<eagle version="7.2.0">
  <drawing>
    <schematic>
      <libraries>
        <library name="library name">
          <packages>
            <package name="package name">
              <smd name="pad name" x= y= dx= dy= layer=/>
              <pad name="pad name" x= y= drill= shape=/>
              <wire x1= y1= x2= y2= width= layer=/>
              <rectangle x1= y1= x2= y2= layer=/>
              <circle x= y= radius= width= layer=/>
            </package>
          </packages>
          <devicesets name="deviceset name">
            <devices>
              <device name="device name" package="package name">
                <connects>
                  <connect gate="gate name" pin="pin name" pad="pad
                      name"/>
                </connects>
              </device>
            </devices>
          </devicesets>
        </library>
      </libraries>
      <parts>
        <part name="part name" library="library
            name" deviceset="deviceset name" device="device name"/>
      </parts>
      <sheets>
        <sheet>
          <instances>
            <instance part="part name" gate="gate name" x= y=/>
          </instances>
          <nets>
            <net name="net name" class=>
              <pinref part="part name" gate="gate name" pin="pin
                  name"/>
            </net>
          </nets>
        </sheet>
      </sheets>
    </schematic>
  </drawing>
</eagle>
```

## 4.4 File Formats

Since Eagle is used as a schematic editor, the Eagle schematic format is used as an input for the 3D electronics. Eagle uses an XML file format that is shown in listing 4.1. The used parts are listed in the `instances` section in `drawing/schematic/sheets/sheet` and are identified by the *part name*. These *part names* can be found in the `parts` section in `drawing/schematic` with some more information about the part such as *library name*, *deviceset name* and *device name*. With this information the part can be found in the `devices` section in `drawing/schematic/libraries/library`. This section provides the *package name* and the assignment of *pin name* and *pad name*. The `package` section in the library contains information about the footprint and the geometrics of the part. The containing `smd` or `pin` pads are identified by the *pad name*. They both have information about the position of the pad. `smd` pads define the dimensions of the pad. `pin` pads define the size of the drilled hole and the shape of the pad. Wires, rectangles, and circles are used to draw the outline of the part. The `nets` section in `drawing/schematic/sheets/sheet` contains the connection nets between the components. Each net is identified by a *net name*. For each member of the net the *part name* and the *pin name* are stored.

Because the Eagle format does not support the necessary data fields for 3D electronics, a new file format was developed. To allow the possibility of editing the Eagle file later, the new format just extends the Eagle format with a new file linked to the Eagle file. With this method it is possible to change, add, and remove parts during the design process. The 3D electronics will also be saved in an XML file which is shown in listing 4.2. The `filename source` describes the *source file* that is used for the design. It will later be possible to use other file types as source file and not just Eagle files. The different `parts` are identified by the *part name*, *library name*, *deviceset name*, *device name* and *package name*. Each `part` stores the *height* of the footprint, the *position* of the component, and its *rotation* around all three axis. `componentsize` stores the size of the component and the `componentpos` stores the position of the component relative to the part's origin. This format can later be extended with information about the placed wires.

Listing 4.2: 3D electronics format

```
<electronics version="1.1">
  <filename source="source file"/>
  <parts>
    <part name="part name" library="library
        name" deviceset="deviceset name" device="device
        name" package="package name">
      <attributes height=/>
      <position X= Y= Z=/>
      <rotation X= Y= Z=/>
      <componentsize X= Y= Z=/>
      <componentpos X= Y= Z=/>
    </part>
  </parts>
</electronics>
```

## 4.5 Data Types

For the implementation, part, and net data types are needed. The data types are all built as a hashmap. That means that the data is stored in key value pairs. The value can be anything for, example a scalar value, a list or a new hashmap. The data types contain all necessary data needed for the 3D electronic design.

Listing 4.3: ElectronicPart

```
part->{name = ,
       library = ,
       deviceset = ,
       device = ,
       package = ,
       height = ,
       volume = ,
       chipVolume = ,
       shown = ,
       printed = ,
       position = (,,),
       rotation = (,,),
       componentsize = (,,),
       componentpos = (,,),
       padlist = (pad->{type = ,
                        pad = ,
                        pin = ,
                        gate = ,
                        drill = ,
                        shape = ,
                        position = (,,),
                        rotation = (,,),
                        size = (,,)}
                  ...)}
```

Listing 4.4: ElectronicNet

```
net->{name = ,
      pinlist = (pin->{part = ,
                       gate = ,
                       pin = }
                 ...)}
```

In this implementation a part represents an electronic component that is used in the design of an electronic circuit. The part data type is shown in listing 4.3. `name`, `library`, `deviceset`, `device` and `package` are extracted from the PCB design source file. The `name` is the identifier for the part. `library` describes the library where the part is from. `deviceset` and `device` contains names to identify the correct part. The `package` contains the package type of the part. `height` contains the thickness of the footprint and will be set during the design process. `volume` and `chipVolume` contain the 3D volumes for the display of the object; they are necessary to identify the part in the 3D scene. The 3D volumes will be generated when they are needed. The value of `shown` shows if the component is currently displayed on the component preview. The value of `printed` is used to identify if a component is already placed during the g-code generation. `position` and `rotation` stores the information for the 3D transformation and are set during the design process. `partsize` contains the size of the main component; it will be extracted from the PCB source file or, if unavailable, from the underling footprint. `partpos` saves the position of the part relative to the origin of the component. `padlist` contains a list of pads, they are an own data type that is also made with a hashmap.

A pad represents a single pin of a component and its counterpart in the footprint. The pad data type contains all information for the pad. The information is completely extracted from the PCB design source file. `type` defines if the pad is a smd pad or a pin pad. `pad` contains the pad name and `pin` refers to the pad identifier that is also used in the net. `gate` contains the gate of the component and is yet not used. `drill` describes the diameter of the hole in a pin pad; it is zero for smd pads. `shape` describes the shape of the pin pad, for example round or octagonal. The shape is currently not used; all pin pads are round. `position` is the relative position of the pad from the origin of the part. `rotation` contains the rotation of the pad. `size` describes the dimensions of the smd pad. It is filled with zeros for pin pads.

An electric net represents a virtual net of connections between parts that will later be realized by wiring. The electronic net data type is shown in listing 4.4. All information of the net data type is extracted from the PCB design source file. The `name` contains the identifier of the net. The `pinlist` is a list of pins, they are an own data type that is also made with a hashmap.

A pin represents a single pin of a component and links the net to the actual component from the part datatype. A `pin` contains a `part` that refers to the name of the connected part. `gate` contains the gate and `pin` identifies the pin of the referenced part.

## 4.6 Implementation

The implementation of the architecture described in section 4.3 is shown in this section. The modifications of the original Slic3r classes are as minor as possible to reduce the effort of integrating changes made to the original Slic3r into this software.

The class `Slic3r::GUI::Plater` has to be modified because the interface for the new 3D electronics has to be integrated here. The modification just contains new buttons and menu entries to start the new interface. This class also holds the schematic file. The new class `Slic3r::GUI::Platter::ObjectElectronicsDialog` contains a window that shows the main panel implemented in the class `Slic3r::GUI::Platter::ElectronicsPanel`, which is the main window for the 3D electronic design. All GUI elements are defined in this class and can be separated into two parts. On the left side of the GUI, all control elements are placed. The right side of the GUI is used to display the 3D model and the electronic components. All 3D objects are shown by the class `Slic3r::GUI::Electronics3DScene` which inherits from `Slic3r::GUI::3DScene` that already exists in the Slic3r. The GUI is described in detail in section 4.8. The class contains the methods to display the object tree. The methods to display the objects in the 3D scene are also defined in this class. There are also methods to display parts and remove parts. The movement by one step can be done with + and - buttons in the interface. For moves on the z axis the step size is always one layer; the step size of the x and y axis is stored in the configuration file. This configuration is hold by this class and is loaded with the class `Slic3r::Config`. It contains some basic settings like the offset for the component, the possibility to set individual chip heights for specific packages, the default extruder for the footprint and the part, and a parameter to deactivate the hiding of parts of higher layers. The object list from the Slic3r cannot be edited because it is used all over the Slic3r. Therefore, the GUI class provides a method to find the part referred to a given volume by iterating over all parts. Any part can be selected through the object tree or by clicking at the object in the 3D scene. The 3D object is shown by the printing toolpath. A slider on the right side of the GUI selects the currently displayed layer. All components that are placed above the currently displayed layer will not be shown. Components can be placed with the mouse, therefore, the mouse handler of `Slic3r::GUI::3DScene` is overloaded in the `Slic3r::GUI::Electronics3DScene` class.

The load and save operations are handled by the `Slic3r::Electronics::Electronics` class. This class just delegates the file read or write operations to the different file readers. Eagle XML files are read by the class `Slic3r::Electronics::Filereaders::Eagle` class. The new 3D electronic XML files are read and written by the `Slic3r::Electronics::Filereaders::3DElectronics` class. Both file readers extract all necessary data and crate a schematic with the provided data types.

The part data type that is defined in the class `Slic3r::Electronics::ElectronicPart` generates parts as described in section 4.5. This class also provides a method to place and remove a part. Each part can output its part data in an XML format to place these components. 3D mesh models for the chip and the footprint are also created in this class. The part class also provides transformation methods due to the three different coordinate systems. The main coordinate system is the print base coordinate system. The object coordinate system is referred to the 3D object and can be rotated around the Z axis. The part coordinate system is refereed to each single part, it can be rotated around all three axis. The transfer function uses the following formula to calculate the position of the object.

$$X_{base} = X_{object} * cos(\theta) + Y_{object} * sin(\theta) + X_{offset} \quad (4.1)$$

$$Y_{base} = X_{object} * -sin(\theta) + Y_{object} * cos(\theta) + Y_{offset} \quad (4.2)$$

The generation of the geometric primitives for the models is done in the class `Slic3r::Electronics::Geometrics`. And converted into triangle meshes and Slic3r models with the methods provided by the classes `Slic3r::TriangeMesh` and `Slic3r::Model`. The class `Slic3r::Electronics::ElectronicPad` is just a pad data type without other methods. The classes `Slic3r::Electronics::ElectronicNet` and `Slic3r::Electronics::ElectronicNetPin` just create their data type without any other methods.

To use the automatic pick and place support developed by [Wasserfall, 2015], the g-code has to be modified at a specific position. To add the component placement to the g-code, the schematic is handed to `Slic3r::Print` which passes it further to the class `Slic3r::Print::GCode`. The g-code command to place a part will be integrated at the end of each layer (listing 4.5). Each part decides if it has to be placed on the just finished layer. All parts that are not placed until the last layer, will be placed at the end of the print. This is necessary to place parts that lay on the top layer of the object.

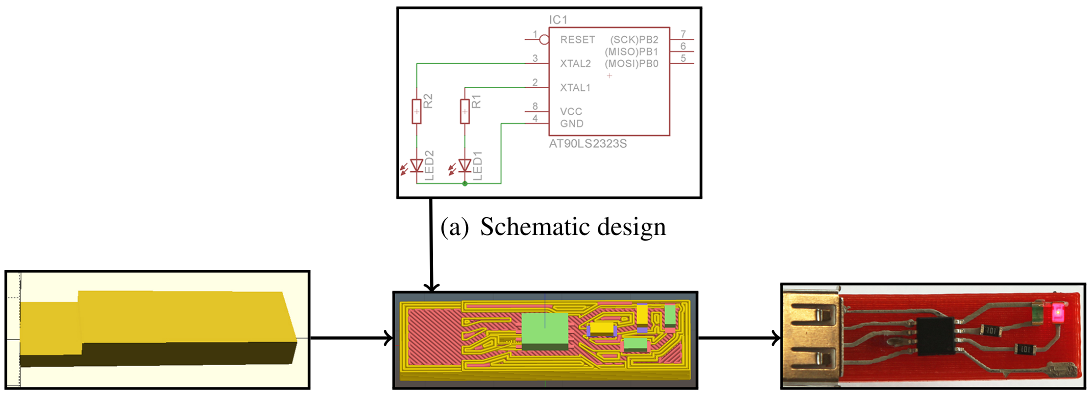
Listing 4.5: Inserting the placement of the parts into the g-code

```
sub process_layer {
  ...
  my $id = 1;
  foreach my $part (@{$schematic->{partlist}}) {
      $gcode .= $part->getPlaceGcode($layer->print_z, $id);
      $id += 1;
  }
}
```

At the end of the g-code, the description for each part is added. This description is generated by each part. The generated g-code is then usable with the vacuum pick and place system developed by [Wasserfall, 2015].

## 4.7 Integration in Design Process

The PCB design process starts with the schematic design of the circuit. In the 3D printing process the design of the 3D object is the first step. The process to design 3D electronics integrated into 3D printable objects starts simultaneously with these two tasks (figure 4.3(a) and 4.3(b)). When the schematic design is done in the PCB design software, for example Eagle, it will be saved as a schematic file. The 3D design of the object is done with a CAD software, such as OpenSCAD and, like for the classical printing process, will be saved as an STL file. The 3D object will then be loaded into the Slic3r and is sliced. When the slicing is done, the integration of the electronics can begin (figure 4.3(c)). Therefore, the schematic file is loaded and the different parts can be placed inside the object. Both, the 3D object and the schematic, can be edited during the design process. Although the current implementation not yet supports wiring, this step would be next. When the slicing part of the software is capable of handling the footprints, components, and wires, the design can be sliced again and printed on an FFF printer with extrusion based direct writing integration (figure 4.3(d)). The finished product of figure 4.3(d) is not produced with the software of this work. It is just used to get a vivid example of how the full process can be used when the missing features are implemented in future works.



(a) Schematic design

(b) Object design

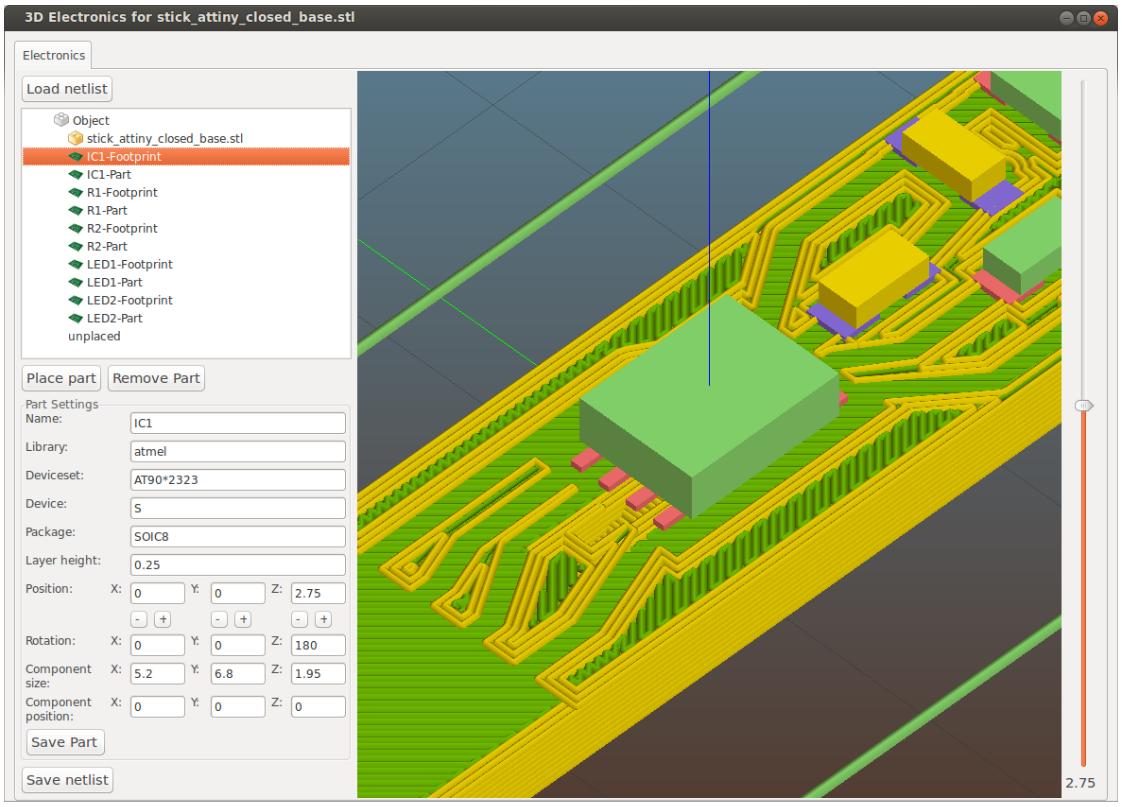
(c) Placement

(d) Production

Figure 4.3: 3D electronics design process

## 4.8 Program Usage

The integration of the electronics, mentioned in the previous section, will be described more in detail in this section. To start the 3D electronic window, the 3D object has to be selected and the 3D electronics button has to be pressed. Now the 3D electronic window, as in figure 4.4, is visible. To start the design process, the previously designed electronic circuit, or an already designed 3D electronic circuit, has to be loaded. This is possible with the `Load Netlist` button. The program loads the configuration and extracts all necessary information. All loaded parts are now displayed in the object tree on the left side. Unplaced parts are listed below the unplaced category. The information for the currently selected part is shown in the `Part Settings` section. Selected parts can be placed in two different ways. The first method is to select a layer for the placement with the slicer on the right side of the window, press the `Place Part` button and click on the layer of the object to place the part there. The other way is to enter coordinates in the part settings section and press the `Save Part` button. With both methods the footprint and the component will be placed in the object. They are shown as two different objects, yet they always represent one single part. That is necessary to



Object
- stick_attiny_closed_base.stl
- IC1-Footprint
- IC1-Part
- R1-Footprint
- R1-Part
- R2-Footprint
- R2-Part
- LED1-Footprint
- LED1-Part
- LED2-Footprint
- LED2-Part

unplaced

Part Settings

| | | | | | | |
|---|---|---|---|---|---|---|
| Name: | IC1 | | | | | |
| Library: | atmel | | | | | |
| Deviceset: | AT90*2323 | | | | | |
| Device: | S | | | | | |
| Package: | SOIC8 | | | | | |
| Layer height: | 0.25 | | | | | |
| Position: | X: | 0 | Y: | 0 | Z: | 2.75 |
| Rotation: | X: | 0 | Y: | 0 | Z: | 180 |
| Component size: | X: | 5.2 | Y: | 6.8 | Z: | 1.95 |
| Component position: | X: | 0 | Y: | 0 | Z: | 0 |

Figure 4.4: 3D electronics main window

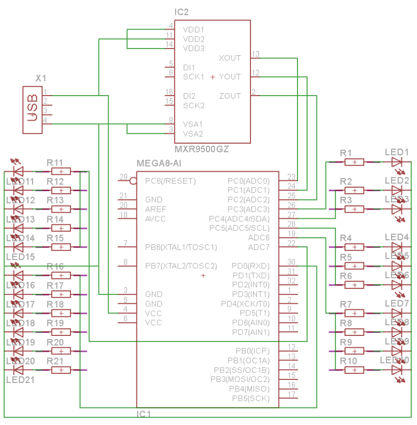
allow the slicer to treat the component and the footprint differently. The footprint will be placed inside the current layer and the component on top of it. All parts can be edited in the `Part Settings` section. `Library`, `Deviceset`, `Device` and `Package` are extracted from the source file. `Layer height` defines the thickness of the footprint which is automatically set to the thickness of the layer where the part is placed on and can also be set by the user. `Position` and `Rotation` represents the position and rotation of the part relative to the origin of the object. `Component size` represents the outline of the part, which is extracted from the source file. `Component position` shows the position of the component relative to the origin of the part. To apply the changed information the `Save Part` button has to be pressed. Parts can be selected either by selecting them in the object tree or clicking on them in the 3D scene. A selected part can be moved by pressing the + and - buttons beside the coordinates, by changing the coordinates and pressing `Save Part`, or by pressing `Place Part` and replacing it with the mouse. To remove a selected part, the `Remove Part` button has to be pressed. Removing a part always removes both, the footprint and the component from the object. To save a design, the button `Save Netlist` has to be pressed. The design will be saved in a 3D electronics schematic file at the same place as the opened PCB design schematic. This is necessary because the 3D electronics file format only saves information that is not present in the PCB design schematic. The source schematic can be changed during the design process. Removed parts will be removed from the 3D electronics too and added parts can be placed. When the design is finished, the 3D electronics window can be closed. The button `Export G-code...` triggers an export dialog. The g-code will be saved in the chosen file and if parts have to be placed, the commands will be added by the implementation to the g-code.

# 5 Evaluation

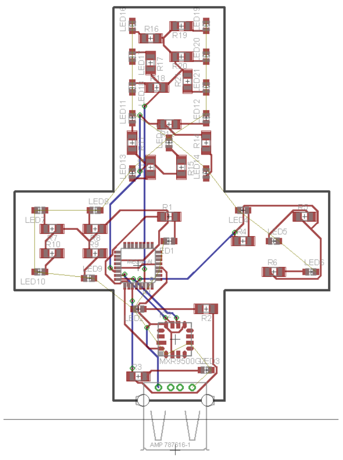
The new design process is great for the integration of electronics into 3D printable objects and works very well with the used setup. The parts can be placed freely in the 3D space and can be rotated around all three axis, so the produced circuits are real 3D electronics. Some simulations and checks from the PCB design tool can still be used. The implemented software in this work is easy to use and can be easily extended because it is an open source software. The design is not very complicated and was feasible within the time limitations of a bachelor thesis. The display of the toolpath and the actual layers are very helpful in the design process. New file formats can be implemented by just adding a new filereader to the source code. The necessary nets for the wiring already exist in the implementation.

## 5.1 Tests

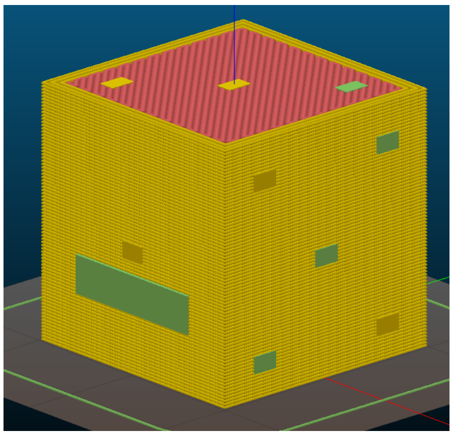
This section shows some example circuits designed with the implemented software. The first test was done to test the implementation of the placement methods and the time that is necessary to do this. The design was done by the author of this work so it might not be fully representative since the author is very well known with the work. However, it shows that the new design process accelerates the production of new 3D electronic devices. For the test a test circuit was designed as a schematic in Eagle (figure 5.1(a)). Then the time it took to place the components on a PCB (figure 5.1(b)) and in a $20mm^3$ 3D cube (figure 5.1(c)) was measured. The test circuit is a dice with LEDs on each side and a gyroscope inside. It is very similar to the circuit designed by [MacDonald et al., 2014]. For example, the dice can light the LEDs up which are on the upper side. The placement of the components without the visible cube is shown in figure 5.1(d). The placement on the PCB took about 17 minutes without the wiring. The wiring only took another minute because the autorouter provided by Eagle was used. The placement in the 3D object took about 23 minutes also without wiring because it is not available yet. Both circuits are just first draft designs. There are four main reasons why designing the 3D circuit took a little bit longer than the PCB version:



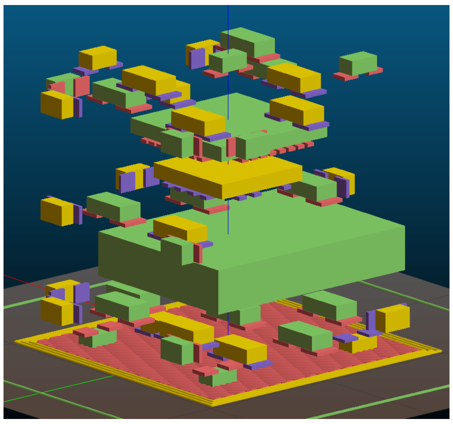
(a) The Eagle schematic



(b) The designed PCB circuit



(c) The designed 3D circuit

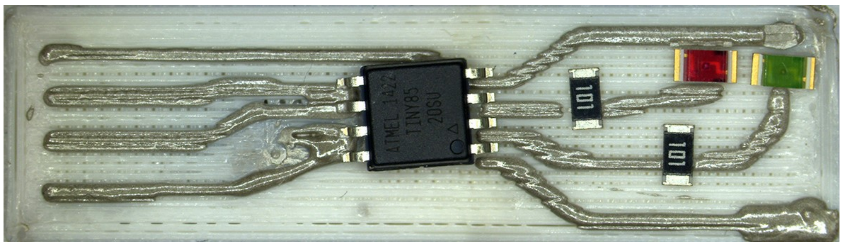


(d) The 3D circuit without the cube

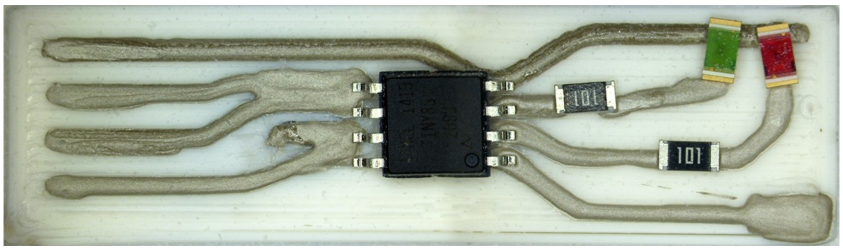
Figure 5.1: Test circuit to test the design performance of the implementation.

- Due to missing rubber bands the user has to look back on the circuit schematic in Eagle to identify components that have to be placed closely together.
- The accurate placement is a little bit complicated because of the missing grid to arrange the components on.
- The turning of the devices around more than one axis is not very handy. The user has to try out different angles to get the desired result.
- The 3D display of the components gets a little bit slow with all components placed inside the object, which was mostly a problem when moving a part with the + and − buttons.

Even with these limitations, a difference of just 6 minutes might be much faster than the design methods that others used before.



(a) The first attempt



(b) The second attempt

Figure 5.2: A test circuit to test the design and placement of the implementation.

A second test aims at the production of a real 3D printed circuit. For the test, the same circuit that is shown in section 4.7 is used. The circuit schematic was again designed in Eagle. Due to the missing wiring in the implementation, the wiring was done with the previously used CAD design approach. The 3D designed part was loaded in the Slic3r and the wiring object was added. Then the components were placed in the implemented 3D electronics window. The whole design was then sliced and transferred to the printer. The printer printed the entire 3D object, added the circuits, and then placed the components on the positions that are set by the design.
In the first print (figure 5.2(a)) the implementation had a bug. All components were placed at the correct position but rotated by 90 degree. The IC in the center of the print was placed correctly because it was placed wrongly on the component tray.
The second print (figure 5.2(b)) was printed with a patched version. In this print all components are placed at the correct position with the correct rotation. Some components are a little bit crooked because the pick and place mechanic had problems picking the components correctly. The wires also have some electrical shorts due to problems with the slicing of the wires. Besides these production related problems, the results are very good. All components are placed on the same position as in the design. With the automated placement the design process is a lot easier than without this implementation although the wiring has been done manually.

## 5.2 Limitations

The main limitation of the design process is the fixed slicing tool. Methods that depend on a different slicing tool can not be used with this method. For now, the implementation is just a prototype. Wiring is not implemented; so the implantation is just usable for the placement of objects and the printing of the footprints. The wiring has to be done manually before the actual placement. The support of design rules and its possibility to check them is not implemented. Like the test section mentions, rubber banding is missing, no positioning grid is available and the rotation is complicated. The 3D display of the Slic3r is slow when many components are placed because the 3D scene has to be completely redrawn every time a component is displayed or hidden. The possibility to hide single components does not exist. The colors of the 3D objects are set randomly so the user can not distinguish the component and the footprint by the color. The pins of symmetrical components can not be identified due to the missing caption of the pins. The integration in the g-code only works correctly when the components are rotated solely around the Z axis. So far, the implementation is just capable of handling a single 3D object with electronics.

# 6 Conclusion

The lack of a suitable software for the design of electronic circuits in 3D printable objects, as it is mentioned by [Lopes et al., 2012], [MacDonald et al., 2014] and [Wasserfall, 2015], is a huge problem. In order to solve this problem, this work shows the existing approaches from different authors and evaluates them. The existing approaches are **CAD Design**, **PCB Design with CAD Support** and **Autodesk Project Wire**. Due to a missing suitable approach, new ones have to be developed. The four new approaches are **Multilayer PCB Design**, **PCB Design Software with 3D Support**, **Independent 3D Electronic Design** and **3D Electronic Design in Slicing Software**. The new approaches are also evaluated. In order to choose the best approach, the requirements of the software are listed. The method that fits the requirements best is the **3D Electronic Design in Slicing Software** method. It is implemented in this work by integrating it into the slicing software Slic3r. The most important arguments to use this approach are the existence of a 3D representation and the possibility to see the toolpath while designing the circuit. For the PCB schematic design Eagle is used because the XML format is easy to use. Then the architecture of the software is defined and the classes are grouped into the categories GUI, file handlers, data types and g-code generation. The used file format is analyzed and a new file format for 3D electronics is developed. The new data types, ElectronicPart and ElectronicNet, that contain all necessary information are described. The software is implemented as an extra window into the Slic3r interface. The design process and the usage of the software is shown. Tests have shown that the used implementation works well and is easy to use. Real 3D circuits can be designed with lower effort than before. The software can be extended easily because of its open source culture. The source code can be found on GitHub at the URL `https://github.com/Zip-o-mat/Slic3r-with-Electronics`. The version corresponding to this work is saved in the branch `bachelor_thesis`.

## 6.1 Outlook

3D printing and especially 3D printed electronics can be part of the next industrial revolution. some day it may be possible to print an entire phone with a 3D printer. [Sun et al., 2013] already showed that it is possible to print a battery. The possibility of 3D printed circuits is shown by [Palmer et al., 2004]. The size of electronic circuits can be shrinked with 3D printed electronics. Maybe special 3D parts will be produced with connections on all sides rather than just on the bottom side of the component. This will lead to smaller components because currently the size of a component is often limited by the confined space for the connections on the bottom side of the component. 3D electronics will reduce the needed space for the circuit with increasing resolution of the printers. 3D printable electronics can also reduce the costs of materials and assembly for prototypes and small series of products. Even new applications we cannot even think of now will be possible with 3D printable electronic.

The development of the design software, like the one in this work, has to go further. The missing features in this implementation, such as the wiring and the modification of the slicer to print circuits properly, have to be added. Even features like wiring with a simple autorouter or the support of design rules can be implemented in future works. Most research effort so far aims to the hardware of 3D printed electronics. All research except Autodesk Project Wire [Autodesk, 2015] only addresses the hardware and gives the software less attention. Autodesk does not publish any information except on their marketing website. To develop the field of 3D printed electronics further, it is necessary that the software and the process is addressed by more researchers.

# Bibliography

[Ammon, 1987] Ammon, P. (1987). *Entwurf von Leiterplatten*. Hüthig, Heidelberg.

[Autodesk, 2015] Autodesk (2015). Autodesk Project Wire - website. http://www.voxel8.co/software/. accessed on 15 Sept. 2015.

[Espalin et al., 2014] Espalin, D., Muse, D., MacDonald, E., and Wicker, R. (2014). 3D Printing multifunctionality: structures with electronics. *The International Journal of Advanced Manufacturing Technology*, 72:963–978.

[Evans, 2012] Evans, B. (2012). *Practical 3D Printers: The Science and Art of 3D Printing*. Apress, Berkely.

[Gibson et al., 2015] Gibson, I., Rosen, D., and Stucker, B. (2015). *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. Springer, New York, 2nd edition.

[Goth et al., 2011] Goth, C., Putzo, S., and Franke, J. (2011). Aerosol Jet printing on rapid prototyping materials for fine pitch electronic applications. In *Electronic Components and Technology Conference*, volume 61, pages 1211–1216.

[Gutierrez et al., 2011] Gutierrez, C., Salas, R., Hernandez, G., Muse, D., Olivas, R., MacDonald, E., Irwin, M., Wicker, R., Newton, M., Church, K., and Zufelt, B. (2011). Cubesat Fabrication through Additive Manufacturing and Micro-Dispensing. *International Conference and Exhibition on Device Packaging*, 7.

[Horvath, 2014] Horvath, J. (2014). *Mastering 3D Printing*. Apress, Berkely, 1st edition.

[Hull, 1988] Hull, C. (1988). Stereolithography: Plastic prototypes from CAD data without tooling. *Modern Casting*, 78:38.

[Jansen, 2001] Jansen, D. (2001). *Handbuch der Electronic Design Automation*. Hanser.

[Lopes et al., 2012] Lopes, A., MacDonald, E., and Wicker, R. (2012). Integrating stereolithography and direct print technologies for 3D structural electronics fabrication. *Rapid Prototyping Journal*, 18:129–143.

[Lopes et al., 2006] Lopes, A., Navarrete, M., Medina, F., Palmer, J., MacDonald, E., and Wicker, R. (2006). Expanding rapid prototyping for electronic systems integration of arbitrary form.

[MacDonald et al., 2014] MacDonald, E., Salas, R., Espalin, D., Perez, M., Aguilera, E., Muse, D., and Wicker, R. (2014). 3D printing for the rapid prototyping of structural electronics. *Access, IEEE*, 2:234–242.

[Medina et al., 2005] Medina, F., Lopes, A., Inamdar, A., Hennessey, R., Palmer, J., Chavez, B., Davis, D., Gallegos, P., and Wicker, R. (2005). Hybrid Manufacturing: Integrating Direct Wire and Stereolithography.

[Navarrete et al., 2007] Navarrete, M., Lopes, A., Acuna, J., Estrada, R., MacDonald, E., Palmer, J., and Wicker, R. (2007). Integrated layered manufacturing of a novel wireless motion sensor system with GPS. Technical report.

[Palmer et al., 2004] Palmer, J., Davis, D., Chavez, B., Gallegos, P., Wicker, R., and F., M. (2004). Rapid Prototyping of High Density Circuitry. *Conference, Rapid Prototyping Association of the Society of Manufacturing Engineers*, 24.

[Perez and Williams, 2013] Perez, B. and Williams, C. (2013). Combining additive manufacturing and direct write for integrated electronics–a review. In *International Solid Freeform Fabrication Symposium*, volume 24, pages 962–979.

[Shemelya et al., 2015] Shemelya, C., Cedillos, F., Aguilera, E., Espalin, D., Muse, D., Wicker, R., and MacDonald, E. (2015). Encapsulated Copper Wire and Copper Mesh Capacitive Sensing for 3D Printing Applications. *Sensors Journal*, 15:1280–1286.

[Sun et al., 2013] Sun, K., Wei, T., Ahn, B., Seo, J., Dillon, S., and Lewis, J. (2013). 3D Printing of Interdigitated Li-Ion Microbattery Architectures. *Advanced Materials*, 25:4539–4543.

[Swensen et al., 2015] Swensen, J., Odhner, L., Araki, B., and Dollar, A. (2015). Injected 3D Electrical Traces in Additive Manufactured Parts with Low Melting Temperature Metals. *Journal of Mechanisms and Robotics*, 15.

[Walker and Lewis, 2012] Walker, B. and Lewis, J. (2012). Reactive silver inks for patterning high-conductivity features at mild temperatures. *Journal of the American Chemical Society*, 134:1419–1421.

[Wasserfall, 2015] Wasserfall, F. (2015). Embedding of SMD populated circuits into FDM printed objects. In *Solid Freeform Fabrication Symposium*, volume 26, pages 180–189, Austin.

# Selbständigkeitserklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internetquellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ich bin mit der Einstellung der Arbeit in den Bestand der Bibliothek des Fachbereichs Informatik einverstanden.

Hamburg, den 10. Dez. 2015

Daniel Ahlers