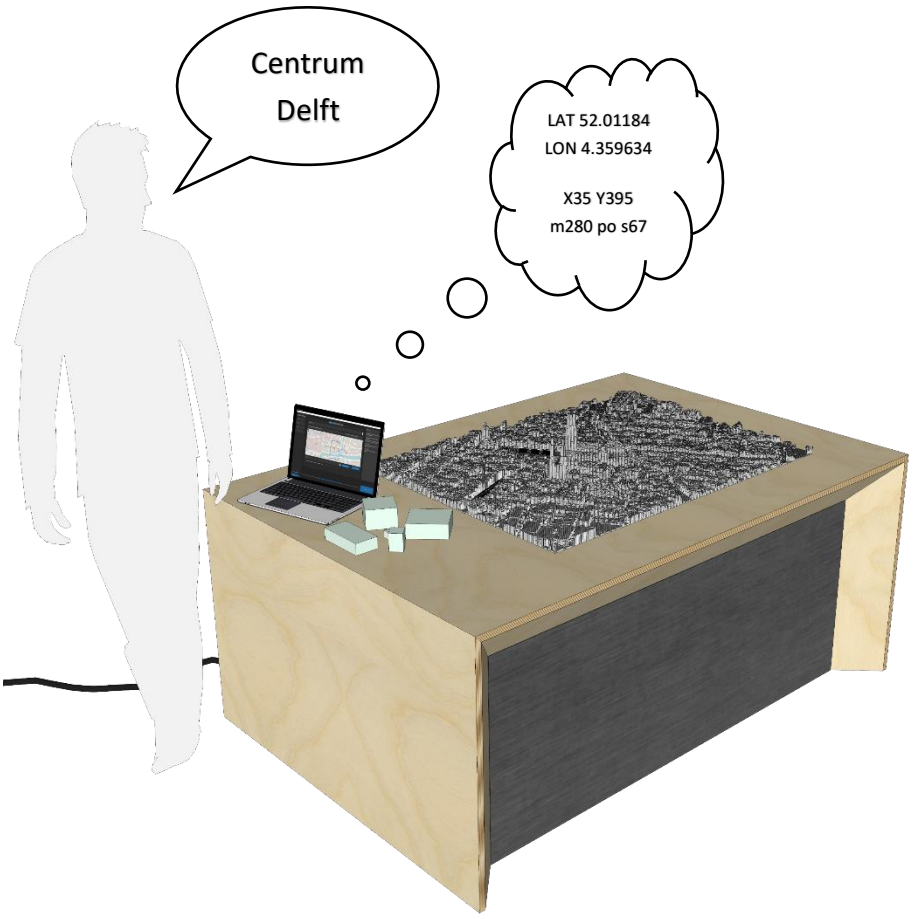
PIN MODEL AS A DESIGN TOOL

Affordable upscaling of pin-type models for physical mass analysis of urban, rural, and complex building sites.



Pin model as a design tool

Affordable upscaling of pin-type models for physical mass analysis of Urban, rural, and complex building sites.

AR3B025 Building Technology graduation studio
MSc Architecture and Built Environment, Delft University of Technology

Cedric Spijksma

5481627
27-06-2023

Serdar Asut (chair of Design Informatics)
Marcel Bilow (chair of Façade & product design)

Keywords:
Pin-type machine, design tool, Urban, Rural, Environment, mass study

Acknowledgment

I want to express my sincere gratitude to my tutors, Serdar Asut and Marcel Billow, for their expertise, enthusiasm, valuable guidance, support, and encouragement throughout my research and project. It helped me overcome various obstacles and challenges, and their constructive feedback was invaluable in shaping my research.

I would also like to express my gratitude to the people and companies that helped to fund and shape my research. This project would not have been possible without their knowledge, expertise, and financial assistance.

I would also like to acknowledge the valuable contributions of my fellow students, who provided me with support, feedback, motivation, and inspiration. Their constructive criticism and insights were extremely helpful in shaping my research.

Finally, I would like to express my gratitude to my family and friends, who have provided me with unwavering support and encouragement throughout my academic journey. Their support, enthusiasm, understanding, and patience have been my greatest source of strength and motivation, and I am grateful for their presence during this process.

Summary

Architects and urban planners still rely on physical models in the design process, particularly for urban contexts. However, developing such models can be time-consuming and resource-intensive, despite the abundance of digital information available. This study investigates the feasibility of using pin-type models, specifically in the concept and design phase of mass modelling and analysis for urban, rural, and complex building sites.

The research uses a combination of literature review, design-based research, and interviews, leading to the discovery of a promising solution. Existing pin-type models in the market or universities either come at a high cost or lack the necessary resolution to generate detailed urban models. However, this study identifies a method that significantly reduces costs while achieving higher pin resolution, although with some compromises in generation time and holding power compared to current models.

The innovative rubber layer, also known as the state layer, plays a crucial role in the machine's advancements. This layer enables the machine to keep pins in place at a lower cost but with lower holding power. Additionally, leveraging technology from the 3D printing and CNC manufacturing domains, the machine incorporates multiple individually moving motors that can rapidly set up to 40 pins in one motion, reaching speeds of up to 1200 pins per minute. Making it possible to generate large pin sets in less then an hour, meaning it would be faster than traditional methods like 3D printing or manual foam cutting.

Furthermore, the research emphasizes the central role of code in this study. Integrating all the necessary steps into a single code can generate an urban area within minutes. The program facilitates a user-friendly interface for selecting locations, automatically downloading height maps from the AHN (Actueel Hoogtebestand Nederland), converting the data into a digital model compatible with the machine, and exporting it as a G-code file. The machine and code also support displaying additional digital information, such as 3D models and images.

Combining the improved machine and optimized code demonstrates the potential to create physical urban context models from digital data at a relatively low cost compared to other devices. With working times of minutes instead of hours or even days, compared to traditional model-making methods.

Although this research shows considerable potential, there are several challenges that demand attention and additional investigation. The ultimate determination of whether the final product will require a $10,000 or $20,000 investment hinges upon the availability and cost of visible pins. Moreover, the current conversion of rotation to Bowden cables is suboptimal, and further enhancements are necessary. Additionally, the code could benefit from optimization to ensure compatibility with slower computers, thereby improving its speed and efficiency. Furthermore, the code should be extended to enhance its compatibility with 3D models, and ideally, with BIM models as well. It is essential that the code incorporates a function enabling the combination and selection of data, facilitating the movement of only the necessary sections of the machine.

With this research, a solid foundation has been established for further expansion and exploration in this field. The successful development of a working machine code paves the way for practical testing and implementation. Although there are some mentioned additions that could enhance the research, the limitations of time prevented their full exploration in this study. Nonetheless, this research opens up exciting possibilities for future advancements and applications in the field.

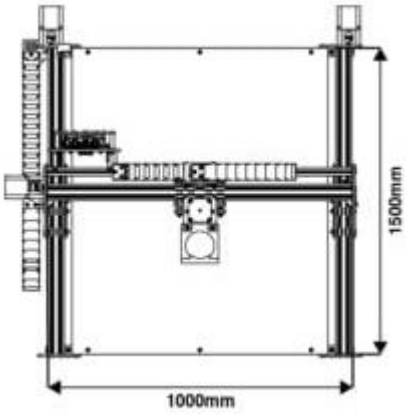
Table of Contents

Glossary

AUTOMAN	- Automated Manufacturing Process Integrated with Intelligent Tooling Systems
BIM	- Building Information Model
CAD	- Computer-Aided Design
CNC	- computer numerical control
CoreXY	- a technique used to move the printhead of a 3D printer or the tool head in CNC machines in the horizontal plane.
DIE	- Removable cast used for fast-changing mass production
EPDM	- Ethylene Propylene Diene Monomer
GUI	- graphical user interface
GRBL	- An open-source, high-performance g-code-parser and CNC milling controller
Matrix	- a number of rows/ columns to form an array/grid
MIT	- Massachusetts Institute of Technology
OSM	- OpenStreetMap
pin type model	- Matrix of pins that can be moved up and down to represent a model
PMMA	- polymethylmethacrylate (Plexiglas, Perspex)
POM	- polyoxymethylene
PSU	- Power Supply Unit
TU/D	- Technical University Delft
TU/E	- Technical University Eindhoven

Technical data

CNC machine used for tests:
Ooznest workbee Z1+ - Ethernet
1000x1500mm (Working Area: 770x1270mm)
Screw driven
duet 2 controller
2500mm/min



Laptop specifications used for code:
model: Lenovo Yogi 9i
operating system: Windows 10 pro
CPU: Intel® Core™ i7-10750h CPU @2.60hz
GPU: NVIDIA GeForce GTX 1650 Ti max-Q
Storage: Samsung 1TB SSD
Screen: 15,6” UHD (3840 x 2160) IPS, VESA400 HDR400 + Dolby Vision™, 500 nits, 72% NTSC
memory: 16GB

1. Introduction

For centuries, designers, architects, and urban planners have used physical models to explore and communicate their ideas in the early stages of a project. These models have numerous applications, from analyzing solar exposure to facilitating the design process.

Physical Architectural and urban/rural models are still important in the early design stage. These models have several functions, as they can be used for mass studies, solar studies, or as part of a design process. Current modeling techniques can be expensive, take several hours to days, and require some skill. (Gibson, Kvan, & Ming, 2002). Some examples can be seen in **Fout! Verwijzingsbron niet gevonden.**.

Hard labour, 3d printing, and CNC milling are currently used to create (large) scale models of areas and building sites. Some research shows the potential of pin models to replace the need for such models. (Follmer, Leithinger, Olwal, Hogge, & Ishii, 2013) Pin-type models are a versatile way to show a model physically, but the current technology does not reach the required pin density. It also does not focus on the use in model making or architecture in general.

How can pin models be used in the concept/design phase of mass modeling and analysis in urban, rural, and complex building sites?

This research aims to investigate the potential of pin models as a physical modeling technique in architectural and urban/rural design. Pin models have the potential to be a versatile and cost-effective option. Further research and testing is necessary to determine their actual capabilities and limitations. To achieve this goal, the following specific objectives have been identified: researching common types of pin-type structures and their current use; identifying the shortcomings of current pin models through a combination of literature review and practical testing; developing a mechanical model/prototype to explore the potential of pin models as a physical modelling technique; creating a software tool to generate pin models digitally; and testing the automated workflow for mass modelling to evaluate the feasibility of using pin models for large-scale urban/rural design projects. Ultimately, this research aims to contribute to the development of more efficient, cost-effective, and sustainable physical modelling techniques for architectural and urban/rural design.

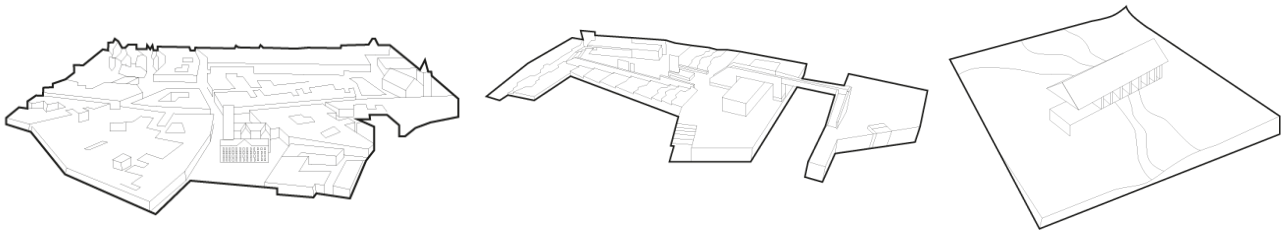


*Figure 1 - city and building models**

* images are based on models made at the department of Architecture at the Technical University Delft

The project started with a literature search on pin-type tooling, moulding, and using models in architecture and urbanism. Scientific papers from the last 20 years were used. Keywords like: pin-type, Pin mould, flexible moulds, reconfigurable models, and adjustable models have been used to find the references in this paper. From these keywords, Universities had their project describing the code and making of these moulds (TU/d, TU/e, MIT, and Stanford). In the next step, interviews were taken with the possible users, like model makers, architects, and engineers.

The second part of the research is based on experiments in a design process. Choosing and altering options and choosing the best solution based on results/data. In this stage, the mechanical model and the code to drive this mechanical machine will be created. A schedule of requirements has been written based on the findings in the Literature search. The decisions were made based on values mentioned in the program of requirements, key values from the literature search, or other knowledge about digital manufacturing like 3D printing, CNC milling, and engineering skills by the writer before the research.

A workflow has been created as a general overview of the project/research. This workflow can be seen in figure 2. This workflow also describes the project's time path within the given timeframe.

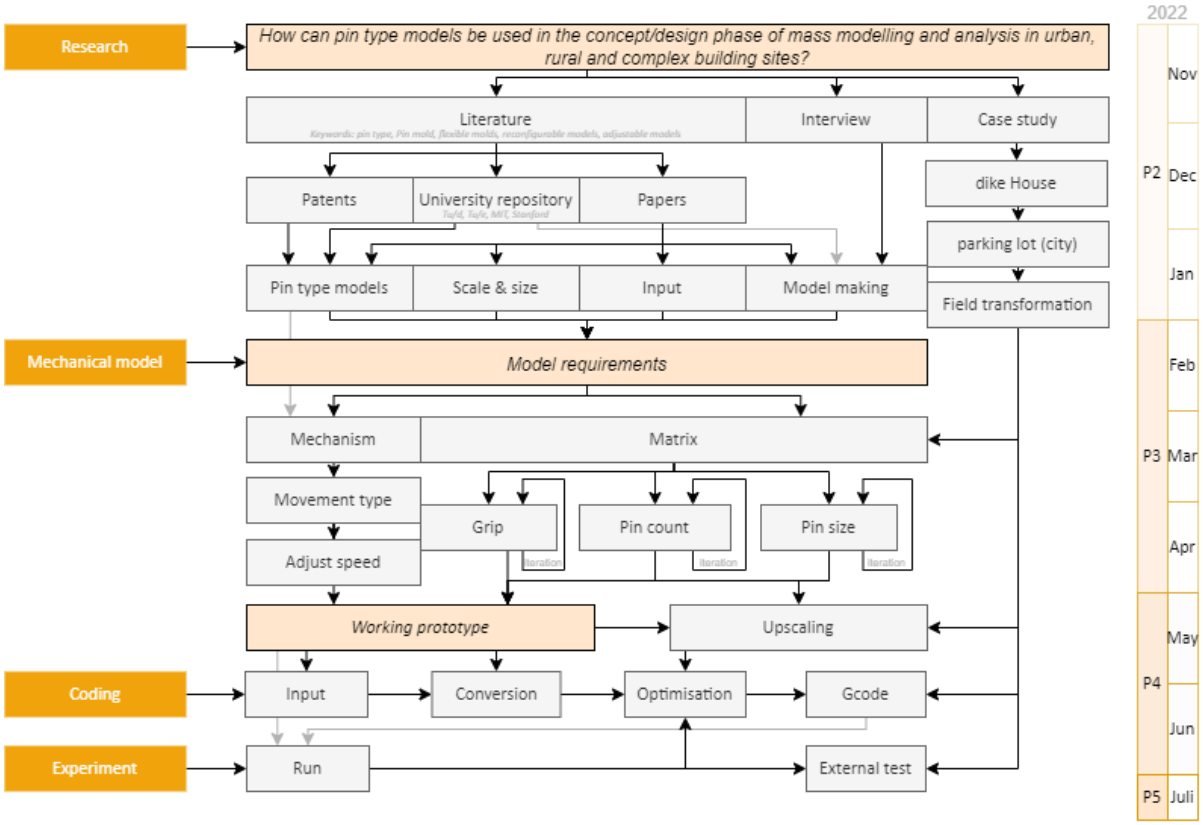


Figure 2 - Methodology and timeframe

Three case studies have been chosen to test the viability of the project. These cases have been based on projects that come across architectural firms. These are not actual cases due to the protection of clients but are very similar to those cases. The first case is the dike house. Here, a complex building site can be seen in the form of a hill on which the building should be placed. The second case study is the urban area, a large-scale city with an old parking spot on which buildings could be placed. The last case study is the rural area, an old farming field that will be converted into housing. The location of these projects can be seen in the table below, and a general overview of the locations in the images below.

Case studies			
nr	**description**	**Location**	**coordinates**
1	dike house	Oudebilddijk, Ouwe Syl, Friesland, Netherlands	53.300303, 5.712277
2	parking lot city	P Acht Zaligheden, Dokkum, Friesland, Netherlands	53.323790, 5.998741
3	Farming land	Aalsumerweg, Dokkum, Friesland, Netherlands	53.334170, 6.003683

Figure 3 - case studies, dike house, parking lot and farm field(left to right)

2. Research

To know why there are no such machines for this application, we must first know what machines are available and their applications. This chapter first looks into the available pin machines, and the second part looks into the design process of architects.

2.1 Movable pin machines

Moveable pins have attracted inventors' and engineers' interest for quite some time, with the first patent from 1863. However, maybe more recognizable to most is the PinPressions™ toy, which became popular in the '80s. Since then, the interest in movable pins has been growing exponentially, shows research by the University of Bologna and the University of Windsor. (Galizia, Elmaraghy, Elmaraghy, Bortolini, & Mora, 2019). Also, the amount of patents shows an exponential growth of interest over the years. (Munro & Walczyk, 2007)

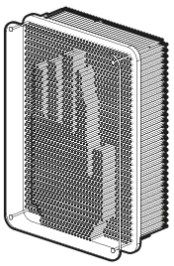


Figure 4 - Pin art

The 2019 published research: the evolution of Molds in Manufacturing from Ridged to Flexible (Galizia, Elmaraghy, Elmaraghy, Bortolini, & Mora, 2019) shows the interest in this topic as well as the options of pin-type tooling. The research looks at academic research as well as industry solutions.

The primary use of these machines is for manufacturing. Products like bottles, forming composite aerostructures, and producing car bodies are mentioned. However, this technology is meant for production lines that would benefit from rapid change.

The research mentions the different options for moving pins. Manual solutions include fixed mould bases, removable dye, replaceable cores and cavities, adjustable inserts, height shims, and exchangeable sub-moulds. They also mention some automated processes, like traveling ejectors, with or without separators, and micro-injection, which uses a dye/fluid to create a curved surface and thermoforming.

The literature search shows all the possible matrix types. As shown in figure 5. There are four main types of matrixes, square uniformly spaced, or tightly packed in a hex or (treaded) round shape. The closed-packed systems are mainly used when high-forming loads are required. The movement of these pins is more complex than the uniformly positioned pins.

The up-and-down movement can be done using several methods. The types described in this research are: manual, mechanical, hydraulic/ pneumatic, and numerically controlled.

The research shows that there is always an additional layer for smoothing for a smooth surface. Elastic, elastomeric, rubber, etc., deformable pin tips, and machining (milling pin top ends or hardenable surface formed by filler) are used for smoothing.

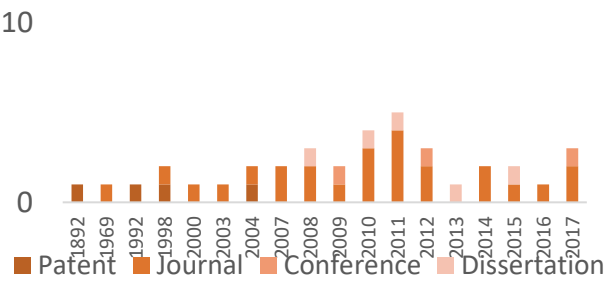


Figure 5 - interest in flexible- and reconfigurable molds (Munro & Walczk)

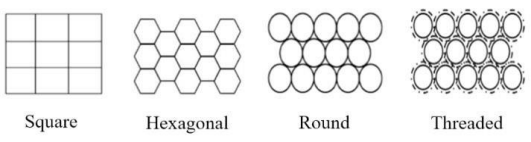


Figure 6 - Pin shape and density classification

The second paper looked at in this research is "Reconfigurable Pin-Type Tooling: A Survey of Prior Art and Reduction to Practice" (Munro & Walczyk, 2007). This paper describes the use of pin-type tooling by looking at the available patents. The use case described in this research defines the industries: of marine, aerospace, automotive, transportation, energy, and architecture as the main uses for such technology. The tool can form, mould, or cast complex curvature parts from metal, polymers, ceramics, wood, and composites are used to create a more permanent mould used in manufacturing or directly to produce a product with high length-to-thickness ratios in manufacturing situations where part variety is great, and production quantities are low.

A Universal tool that can eliminate the need for dedicated tooling (Munro & Walczyk, 2007)

Munro & Walczyk used their knowledge and research to define all the pin-type models and patents into different categories. These categories are defined as:
Pin density: closely packed or uniformly spaced.
Pin actuation methods: mechanical, pneumatic, hydraulic, and use of a robotic manipulator.
Pin Position Control Method: Pin positions can be controlled: manually, automatically (serial or parallel)
Tool Surface Smoothing Method: deformable interpolating layer, deformable pin tips, and covering with a hardenable surface.
Degrees of Freedom: a 2-D row of pins to provide a variable profile, a 3-D matrix of pins to provide a variable surface, and a 4-D matrix of pins with a surface that can be varied in real-time.
Tool Use: A pin-type tool can be classified by how the tool's surface will be used. It can be either direct or indirect. The direct version pins will be driven directly by the source, and a Die cast or additional pins will drive the indirect version.

The ideal tool would consist of a continuous morphing surface based on shape information from a computer model.
Unfortunately, no such technology currently exists. Reconfigurable tooling is currently limited to the discrete element or pins actuated through some automatic means. (Munro & Walczyk, 2007)

To be an ideal tool for moulding, Munro & Walczyk decided on the following points: 1. The tool needs a high surface resolution. More pins mean more detail and product variety; more pins require less surface smoothing. 2. Smooth forming surface. 3. Rapid configuration to utilize most of the machine's capabilities and quickly adapt to a new case. 4. Individual control. 5. Withstanding loads, the tool could be used in different manufacturing techniques, from vacuum forming to bending. 6. Withstand heat. For productions where molten metals or glass are used. 7. Allowance for vacuum. 8. Portable and lightweight.

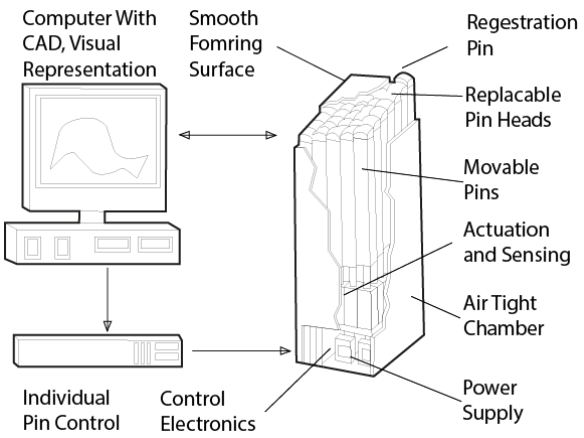


Figure 7 - Characteristics of an ideal reconfigurable tool (Munro & Walczyk, 2007)

The research describes all the patents used for pin tip tooling from 1863 to 2007. These patents have been categorized. This table only shows the computer programmable patents.

Patents			Reconfigurable Tool Taxonomy					Ranking of Characteristics 1 = Bad, 5 = Good					
			Pin density	pin actuation method	pin position control method	tool surface smooth method	degrees of freedom	surface resolution	Smooth surface	rapid reconfiguration	individual pos en ver configuration	easily configurable	withstand high-forming load
Inventor	Year	reference	T1	T2	T3	T4	T5	R1	R2	R3	R4	R5	R6
Whitacre	1971	(U.S. Patentnr. 3,559,450, 1971)	C	R	AS	PT	3-D	4	4	2		3	5
More	1993	(U.S. Patentnr. 5,187,969, 1993)	C	M	AP	PT	2-D	3	2	4	4	2	4
Hoffman	1998	(U.S. Patentnr. 5,846,464, 1998)	U	R	AS	DP	3-D	2	4	3	3	2	
Umetsu	1993	(U.S. Patentnr. 5,192,560, 1993)	C	M	AS	PT	3-D	4		3	2	3	2
Todorok	1993	(U.S. Patentnr. 5,253,176, 1993)	C	R	AS	PT	3-D	3	2	2	2	2	2
Berteau	1994	(U.S. Patentnr. 5,330,343, 1994)	C	M	AS	ILD	3-D	4	4	3	4	3	4
Schroeder	1998	(U.S. Patentnr. 5,738,345, 1998)	C	M	AS	CHL	3-D	4	4	3	2	4	4
Laskowsk	1998	(U.S. Patentnr. 5,796,620, 1998)	U	M	AS	PT	3-D	2		3	3	2	2
Haas	1996	(U.S. Patentnr. 5,546,784, 1996)	C	M	AP	ILD	3-D	3	4	3	3	4	4
Sullivan	2000	(U.S. Patentnr. 6,012,314, 2000)	C	M	AP	ILD	4-D	3	4	5	5	4	4
Haas	2000	(U.S. Patentnr. 6,089,061, 2000)	C	M	AP	ILD	3-D	3	4	3	3	4	4
Papazian	2001	(U.S. Patentnr. 6,209,380, 2001)	C	M	AP	DP	3-D	3	4	3	3	4	4
Papazian	2002	(U.S. Patentnr. 6,363,767, 2002)	C	PH	AS	ILD	2-D	2	4	4	3		4
Haas	2003	(U.S. Patentnr. 6,578,399, 2003)	C	M	AP	PT	3-D	3	2	3	3	4	4
1) C = Closed Packed Matrix and U = Uniformily Spaced													
2) Actuation Method P = Pneumatic H = Hydraulic M = Mechanical, R = Robot or Numerical Control													
3) AP = Automatic Parallel and AS = Automatic Serial.													
4) ILD = lnterpolatinq Laver Detached, ILA = lnterpolatinq Laver Attached, DP = Deformable Pin Tips, CHL = Covered withHardenable Laver and PT = Pins Themselves													
5) 2-D = (x.y), 3-D = (x.y.z), and 4-D = (x,y,z,time)													

Figure 8 - Evaluation of reconfigurable pin-type tooling patents listed chronologically according to tool taxonomy and ideal tool (Munro & Walczyk, 2007)

One of the examples on the next page, the Discrete mould, also comes with extensive research into different pin-type models. (Boers, 2006) in this paper, common aspects and features mentioned are: Large Scale gives coarse resolution, Pins with a square cross-section and hemispherical heads are commonly used in combination with hydraulically actuated cylinders, electronic positioning modules, a numerical milling machine, or manually can be used for movement. Fixation is done on friction, generated using bolts, wedges, hydraulically actuated cylinders, or large spindle mechanisms with individual electronic motor modules.

The three described researches have a lot in common about the available technologies. Actuation methods and smoothing options are all described the same, and no great alternatives are added in one of the papers. Some differences can be found in the definition of a uniformly spaced grid. The first research describes uniformly spaced the same as a square grid, whereas in the second research, uniformly spaced is used for a grid where the pins are apart from each other but within the same space in between. The closely packed is in the first research the same as the staggered hexagon and round principle, as the second research describes closely packed as pins against each other.

The paper on reconfigurable pin-type tooling adds more categorization to the topic, adding the degrees of freedom and tool use. Due to the extensive explanation and categorization, these categorizations could be a beginning in understanding and defining pin-type tools.

The same paper describes the ideal machine as: "a continuous morphing surface based on shape information from a computer model." This definition of the ideal machine would conflict with a pin-type machine used for urban/rural and complex building sites. Cities and building sites do not have a morphed surface but would consist of complex geometry and sudden "straight walls."

Now that we have seen the theoretical part, how do these machines look and work? There are some examples to overview the current technology and its workings.

2.1.1 AUTOMAN

As described in the paper from the University of Bologna and the University of Windsor, also seen in chapter 2.1 Research. (Galizia, Elmaraghy, Elmaraghy, Bortolini, & Mora, 2019). Automated Manufacturing Process Integrated with Intelligent Tooling Systems (AUTOMAN)was developed by the Delft University of Technology (TU Delft, Delft, The Netherlands) and the University of Birmingham (Birmingham, UK) in collaboration with several industrial companies (2014-2017). The aim was to develop the world's first fully reconfigurable pin-based tooling system with in-process sensing and computer control.

Figure 9 shows that the system uses pins to move a pad that would form the metal sheets into the required shape. Most of these machines use a single actuator per pin, which has a high purchasing cost. The pins can go from a couple of hundreds in an evenly spaced grid to a thousand with a closely packed system.

2.1.2 Discrete mold TU/e

As seen in the AUTOMAN project single, pin actuation can be expensive for forming uniform surfaces. Research at the Technical University of Eindhoven (TU/e) has devised a solution to replace the single-pin motion. Divider plates between the pins will keep them in place with force during forming. The pressure will be relieved for setting the pins, and a rocking motor will individually set them to their height. (Boers, 2006). Figure 10 shows the prototype of this discrete mould with 256 pins.

2.1.3 CIKONI Dynapixel

Cikoni is a German company based in Stuttgart. The company focuses on creating moulds in the composite industry. In 2019 they made the Dynapixel. As can be seen in figure 11, the machine has a densely packed pin structure and is driven by treaded pins. The information available about the product is limited.

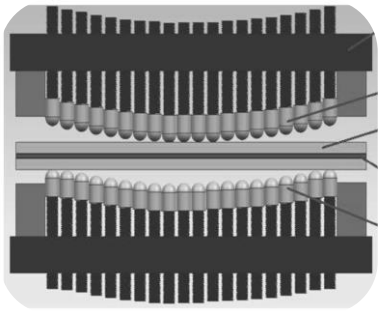


Figure 9 - AUTOMAN system (Ultra Precision Motion, 2015)

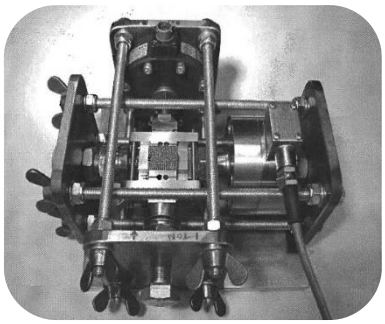


Figure 10 - Discrete mold (Tu/e)

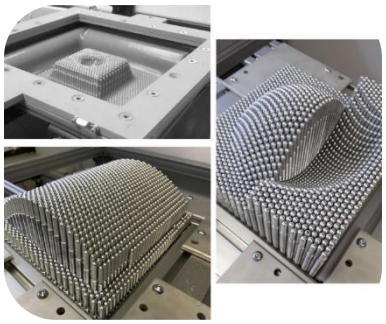


Figure 11 - CIKONI Dynapixel

2.1.4 InFORM (MIT)

Researchers at the Massachusetts Institute of Technology (MIT) have worked on creating a 4D pin-type structure. (Follmer, Leithinger, Olwal, Hogge, & Ishii, 2013) In figure 12, The machine can be seen. The device makes use of individually programmable actuators linked to pins. Using a Kinect sensor, real-time movement can be generated based on hand movement. Using a project, colour can be added to the model. Since the invention of the model in 2013, the machine has been used in over ten different projects. These projects have other use cases and movements, e.g., displaying music waves, human interaction on distance, or object feeling.

The Inform has over 900 pins and costs over €22.500,- in materials to produce. (ORF, 2013) besides the construction cost, the machine takes quite a lot of power. With a max of 2700W, It can use the same energy as a single induction cooking stove. Each actuator uses up to 3W; the actual use has been measured up to 700W. Keeping the display the same still requires up to 300w of power.

2.1.5 Shapeshift (Stanford University)

A different approach to the 4D pin matrix is the shapeshift. Stanford University released this project (Siu, Gonzalez, Yuan, Ginsberg, & Follmer, 2018). In this model, 288 pins make up the surface. By driving the model around based on movement, the model can be extended to represent a bigger object. With this research, a touchable model was thought. Also, with this project, different studies at the university used the machine for their purpose. To this moment, four papers have been written about this project.

Motors with a screw drive the pins in the model. Every pin is individually changeable in real-time by adding microcontrollers to every row. The grid consists of 288 pins. The machine is shown in Figure 13. The cost of the machine would have been in the thousands to produce the prototype. According to the creators, electrostatic adhesive brakes could bring the devices back to €300- dollars a piece. (explained in chapter 2.1.6) (WILSON, 2018)

2.1.6 Electrostatic Adhesive Brakes

As mentioned in the previous chapter, electrostatic adhesive brakes could immensely lower the cost/pin. With this research, a pin could cost around €0,10 apiece. (Zhang & Follmer, 2018) This model makes use of a platform that lowers the pins. In this process, individual pins will be blocked from dropping by applying an electrical current. See figure 14.

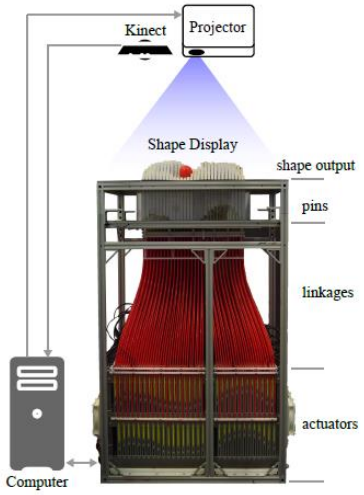


Figure 12 - InFORM (image by Follmer S.)

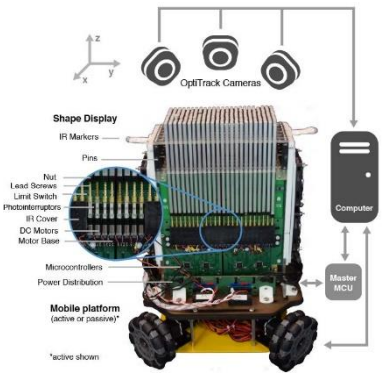


Figure 12 - Shapeshift (figure by Alexa F.)

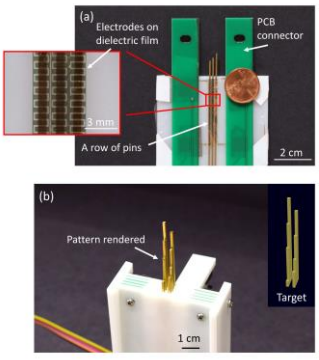


Figure 14 - Electrostatic adhesive brakes)

2.2 Design tool

Models have been used for centuries to design and display. Even now, models are still used to display a project. However, Architects, designers, and engineers also use them for their thinking process. Models can be used in different stages of the design process, from concept to realistic models.

> *Architects use models as a thinking and defining mechanism for understanding and presenting architectural ideas (Smith, 2004)*

Some studies show that models could bridge the mind and the physical. It can help to describe your ideas to others or map your thoughts. (Pallasmaa, 2009). The own thoughts can be seen in easily reading the form, shape, ratios, measurements, and relations between components. The model can show a designer's imagination, giving the feeling of what would be possible. The model can also give back this imagination to the reader. They can evoke memories, dreams, and other visions, creating a dialogue between the designer and the reader.

An example from Mieke Vink en Peter Koorstra (Vink & Koorstra, 2020): Designing on a city square. Masses are placed on a centralized model of a city square. The first model is placed in the center of the square, dividing the square into two smaller squares. A conclusion that can easily be taken. The second model is raised on the centralized model, creating a travel area underneath the building. The masses are quickly changed and tested again, removing parts of the mass to adjust for the movement between what could be two (new) squares. Pallasmaa says the model should be abstract, creating the project's guidelines. Using abstract ways to express the designer's thoughts grants access to the designer's instincts, using the eyes and mind to connect mental and physical connections knowledge. Furthermore, the model must add to an existing model/area. This difference in surroundings and the added model should be clearly visible.

Model making is still a considerable part of architectural universities/academics teaching. And most universities still make use of models in their research and presentations. As also can be seen in the introduction of this paper.

The Faculty of Architecture and Building sciences at the TU/d has a dedicated model-making department named "Form Studies." This department provides students with knowledge, classes, and all the supplies and tools needed to work with architectural and urban models. Professors at the Department have also published several papers regarding the need for these models in the learning and design process. The 2020 published paper of Mieke Vink en Peter Koorstra (Vink & Koorstra, 2020) clearly shows the use of these models. The models are used as a way of thinking. Learning to make, think, interpret, and imagine. You start with a general idea forming the main principles of your design; with changes, new ideas can come quicker, and "mistakes" can lead to new designs/art forms. The students do not have to worry about technical challenges at this stage. Going up a few scales and the technical difficulties will be solved. This scale progression can also be done with the models. How will the model be built? How does it stand up? How is it connected? All these answers can be found in the building of a model.

Another study by the same author from the University of Delft describes new model-making techniques. (Stellingwerff & Koorstra, 2013) The paper describes the problems with new manufacturing techniques and the use of CAD/BIM. The main problem described is that the connection with scale is lost, Details are looked at in a too-early stage, creativity is lost due to pre-defined program settings, and models are made too late. It is only used to show off. When the model is printed, students

discover their design's problems and thus no longer have time to change them. Sketches only present one side of a building. Looking at it from different angles is hard or impossible.

Model and scale are the instruments architects can use to envision their design ideas.

Another study was done In 2020 at the Gdansk University of Technology, Poland. (Agnieszka, Taraszkiewicz, & Taraszkiewicz, 2020) In this study, they did questionnaires about using models in the design process in the first year and asked the same students in the second year if they still use models for their decisions and presentations.

Paper models helped us to grasp the scale and understand how something works.
citated (Agnieszka, Taraszkiewicz, & Taraszkiewicz, 2020)

Traditional model-making is indispensable to feeling how structures should be built. A traditional model allows you to imagine which construction should be used. With a mock-up, it is easier to imagine what you would like to achieve.
citated (Agnieszka, Taraszkiewicz, & Taraszkiewicz, 2020)

Now everything is done digitally, and the amount of content makes remembering difficult. I have no idea what happens on the slides shown for most subjects, but I remember my experiences with physical models much better.
citated (Agnieszka, Taraszkiewicz, & Taraszkiewicz, 2020)

As we have just seen, physical models are still important and can significantly help design and teaching. Even though new model-making methods like CNC milling and 3D printing can help, they are mainly used too late or do not contribute to the design process. Also, note mentioning factors in using these physical models are the abstraction level and the material used.

2.3 Users

Pin models are primarily used in manufacturing. However, they have also found applications in display and design, as discussed in the chapter on movable pin machines. These machines have wide-ranging uses, although no real use cases in the built environment have been described thus far. This chapter explores the various possibilities of pin-type machines in the built environment.

One of the most beneficial fields for pin-type machines is architecture, especially for those involved in planning buildings in urban areas or on complex sites such as hills or mountains. Architectural students and teachers can also utilize this technology, allowing them to adapt the area to their projects easily.

In practice, these are some use cases of how it could be used in the field:
1. Initial research: When a company receives a project, they must gather information about the location, such as existing buildings, surroundings, area history, and local codes. Pin machines can enhance this search by selecting the project area and providing a 3D representation of the surroundings within minutes. Users can observe available space, ground elevations, and neighboring buildings' shapes and roof types. A visualization of such a 3D model can be seen in Figure 15.

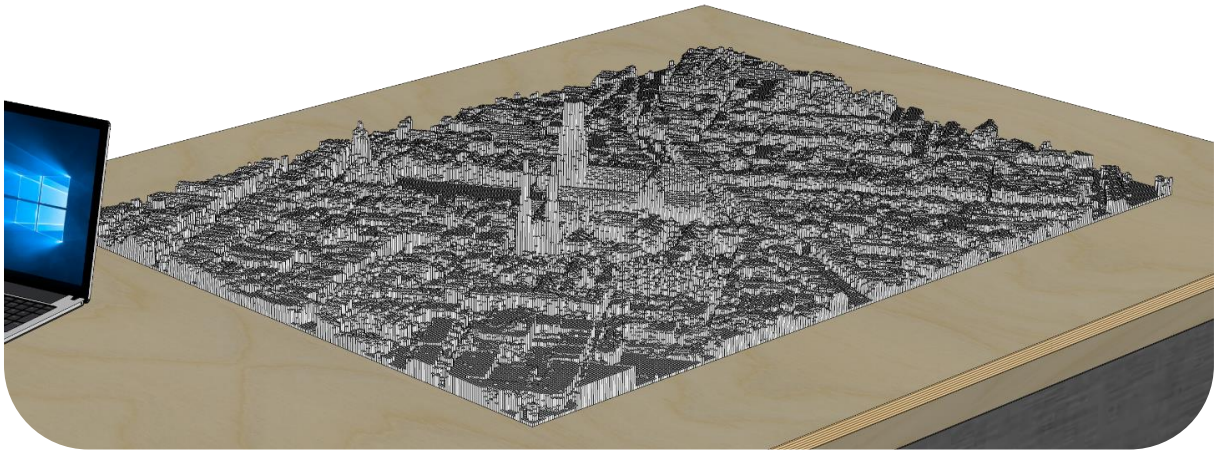


Figure 13 - 3D model Surroundings

2. Mass study: Once all the necessary information is gathered, designers/architects begin working on a rough shape of the building, also known as a mass study. Pin models allow them to utilize the surrounding area without having to model it or rely on external resources manually. The global layout of the area is readily available, enabling architects to place the masses on top of the machine, using foam or Lego, for example. Figure 16 displays such a mass study using foam blocks on top of the machine, changing there shape and effect on the area. This process has also been described in chapter 2.2.

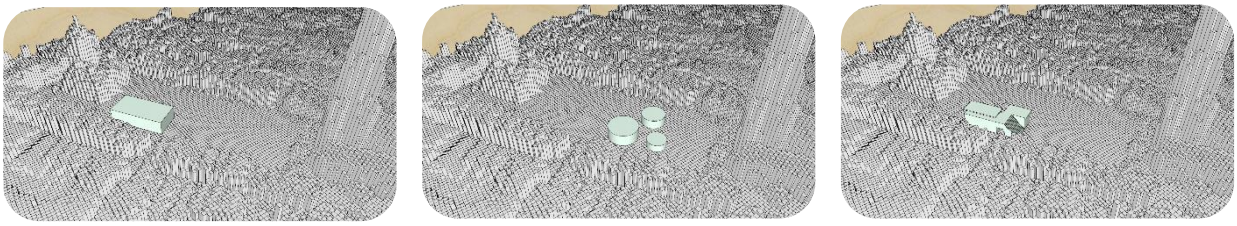


Figure 14 - examples of mass study on top of pin model

3. Shade analysis: After a design is created, it is essential to assess its potential impact on the surrounding environment, particularly in terms of shading effects resulting from the building's height or placement. Pin models can simulate shade by using the masses on top or even by loading a 3D model into the software and generating it inside the pin model (not yet implemented in this research). The same software can also simulate the sun's position and even show the sun's path. Making it able to look at it from different angles with your clients. Figure 17 shows the light bulb going around the model, casting a shadow on the model.

Figure 17 - example shade analysis

Pin machines can also benefit the field of interior design. In kitchen design, for example, products already utilize screens and 3D models to lay out kitchens, like the Kitchenplannertable (Kitchenplannertable, sd), as seen in figure 18. By incorporating pin machines, designers can extend this view, displaying walls in 3D and showcasing the heights of windows and door openings based on their models. This allows for direct interaction with clients and the ability to experiment with furniture placements for better designs. An example of this can be seen in figure 19. 3D printed elements are stored on the outer brim of the table and placed on the generated model. 2 examples of this can be seen on the right of figure 19.

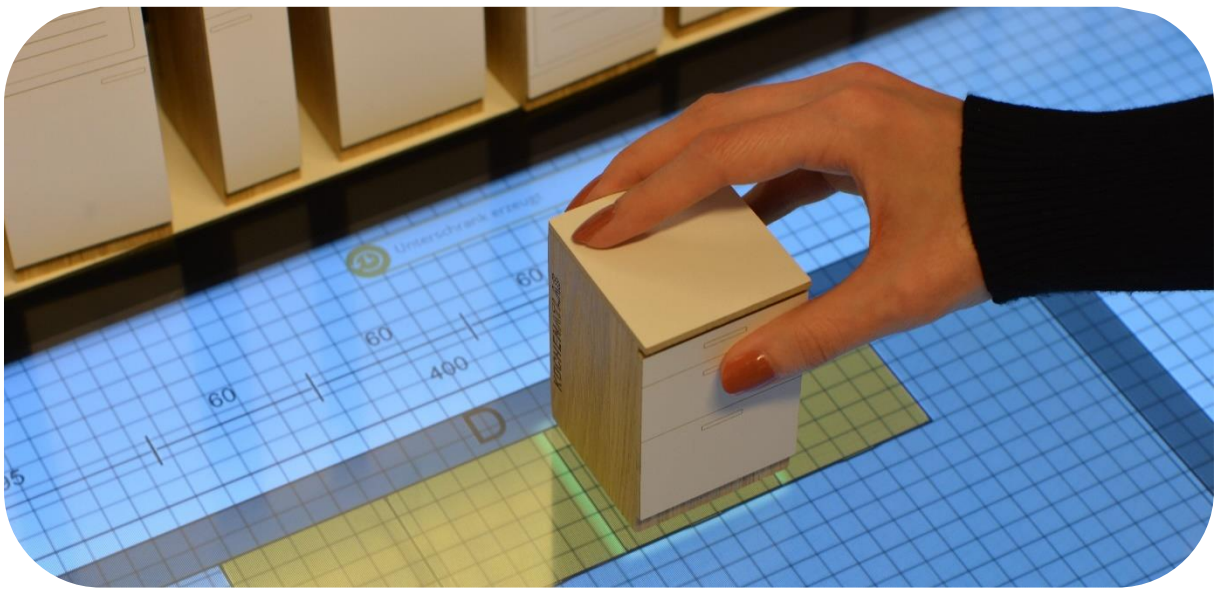


Figure 18 - kitchenplannertable

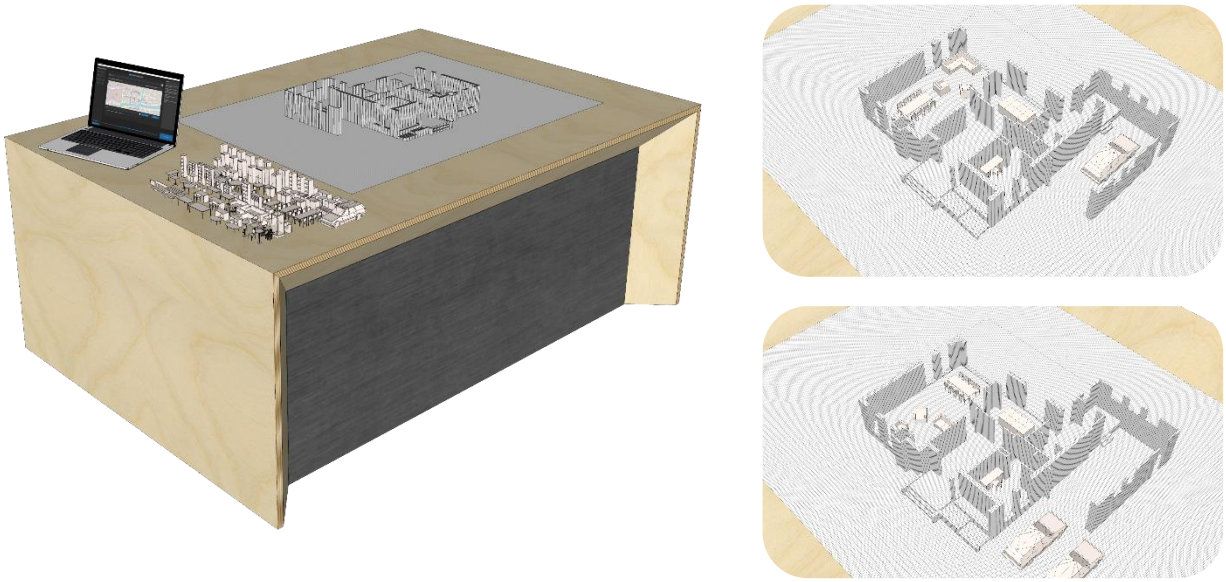


Figure 1915 - interior design pin model

In addition to architecture, pin machines can be helpful in landscaping. Landscape architects can benefit from pin machines to obtain a more accurate understanding of the current ground level in an area. The software can display only the ground elevation, making it easier to visualize height differences in the terrain. This advantage over 2D images is crucial for displaying heights effectively. Landscapers can manually adjust the pins to change the design or utilize other models, similar to the mass study approach in architecture.

Pin machines can assist not only architects and landscape architects but also urban planners in comprehending the current situation of a city. Urban planners can incorporate their models on top of the existing pin model or manipulate the pins to observe the effects of removing buildings and other structures. This provides valuable insights for decision-making processes in urban planning. Figures 15, 16 and 17 have already shown the possibilities for urban design, but this can be displayed more traditionally by only displaying the shapes, as seen in figure 20.

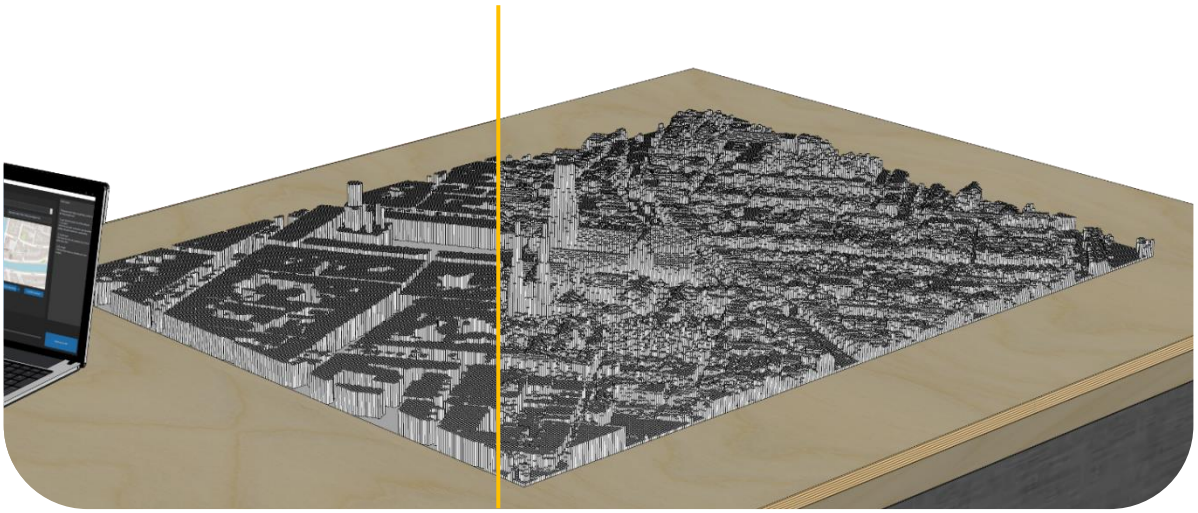


Figure 20 - abstraction

Governments and municipalities can also utilize pin machines to model various city planning and development scenarios. Project developers can create more accurate and detailed models of their buildings and structures, facilitating better visualization and informed decision-making. Figure 21 displays such a case, in which a part of the city is generated, and with a detailed model, in this case, the church, an exposition can be held.

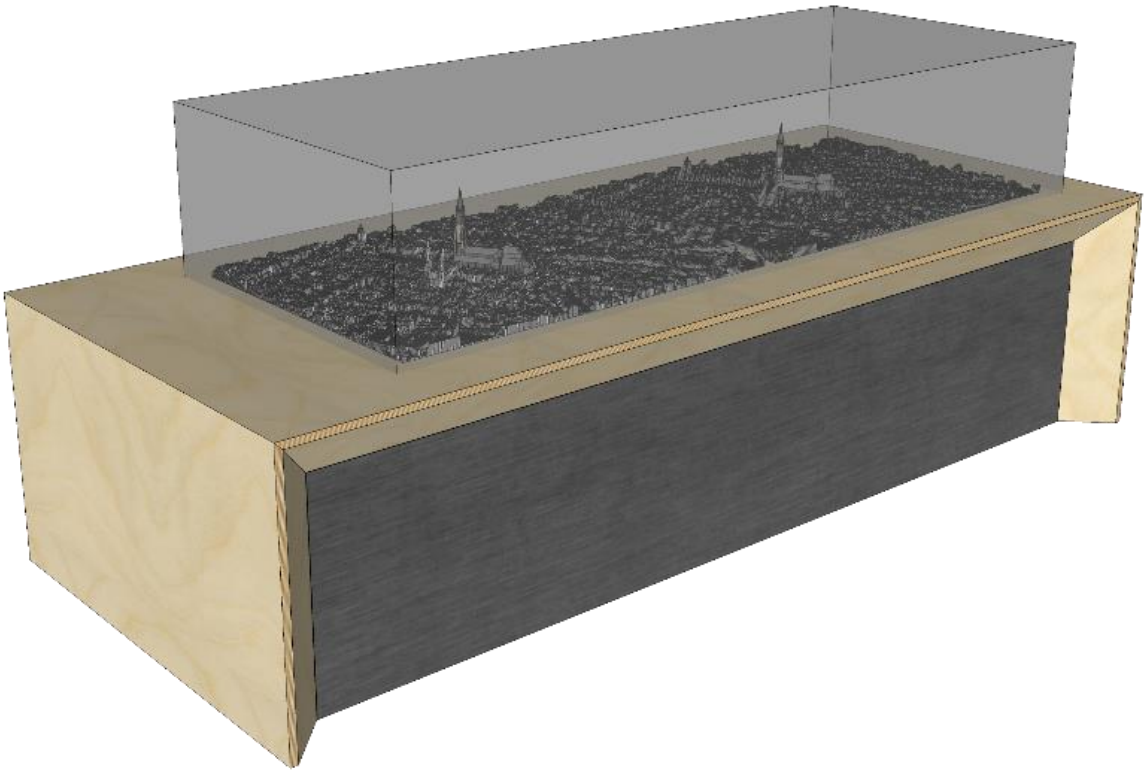


Figure 21 - pin model, display case

While numerous potential applications exist for machine-assisted model-making, further exploration is needed. This research primarily focuses on the architectural and urban planning applications of pin machines in the built environment. As seen with the potential uses displayed in this chapter, there would undoubtedly be a suitable application.

2.4 Input, data acquirement

The central part of this research is about its use in architecture and urbanism. The most considerable input is the location. The region can be found in available digital data to generate the surface when the site is known. This chapter will look at the available data types that could be used for this generation.

City data and building data are regularly available online. The most commonly used data can be found on google maps (Google, 2022), or OpenStreetMap(OSM) (OpenStreetMap, 2022). These databases have data on most of the roads and building plots. Besides that, some cities have building height data, and others even have ground level. The data available on the case studies using OSM can be seen below in figure 22.

Figure 22 - OSM case study locations

In the Netherlands, other databases can be used for accessing building data. Companies used Bag data, which stores information about the built environment and the plots. Available information varies from building date, function, and owner up to the buildings dimensions, gutter height, roof height, etc. (Kadaster, sd)

Besides the BAG data, the institute also posted another dataset, the point cloud data, by AHN. AHN Is a big point cloud made in the Netherlands. This method shows 4 points per m2 generating the surface. The case studies using AHN can be seen in the figure below, figure 23.

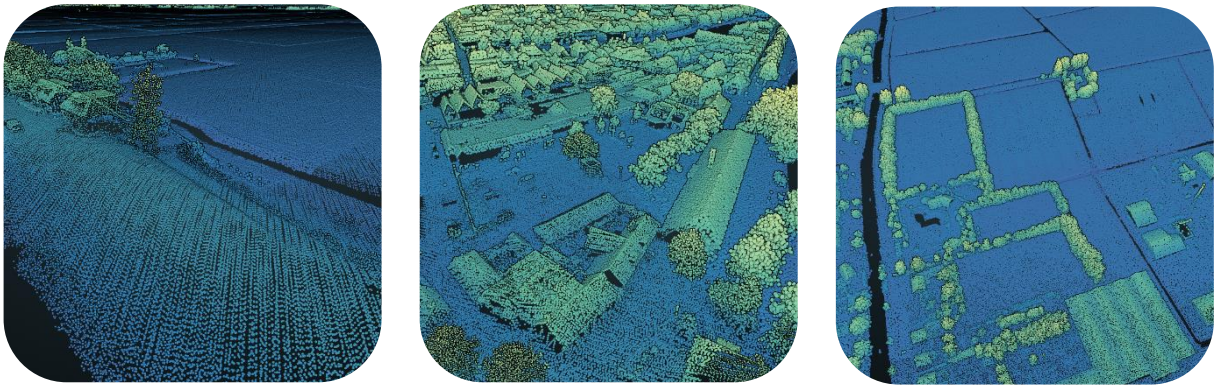


Figure 23 - AHN case study locations

Some areas don't have environmental data available, or sometimes you want an edited version to be used for your model; in this case, manual input should still be possible.

2.5 Size and Scale

Urban models can become massive models. The most prominent model, the Panorama of the City of New York, is 867.2m2. (Bergstrom, 2022) This model fills a complete building and thus is not practical for anyone to use. In this chapter, Scale and size will be looked at.

The model given in the introduction is an extreme case where a model's primary purpose is to showcase a city's size. These sizes are too big for a typical firm and hard to model. All parts of a model should be reachable to work with it as a design tool. For human dimensions, the book Human "Dimension and Interior Space" by Panero (Panero & Zelnik, 1979) can give some great examples of dimensioning. In figure 24 can be seen that a table should have a depth of between 91,4cm and 137.2cm. Models are also not always square due to the printing and modeling on the computer. These sizes also play a role in most cases. An A0 paper, for example, has a size of 841 x 1189 mm. Digital screens don't have a specific size, but most use a width-length ratio of 16:9 and have a screen resolution of 1920x1080 pixels (1080p).

The model uses scale to represent the natural world in a smaller model. This size reduction is the scale on which a model is made. Every building stage has its scale, as seen in the table below. (TU Delft, 2015) for this project, the scales 1:100 up to 1:1000 are important.

Urban/rural	1:1000 - 1:10000
Situation	1:200 - 1000
Overview	1:200
Building	1:50-1:100
Fragment	1:20
construction parts	1:10
Fragment	1.1-1:10

A	182.9–243.8 cm
B	45.7–61.0 cm
C	20.3–30.5 cm
D	50.8–61.0 cm
E	91.4–121.9 cm
F	182.9–259.1 cm
G	91.4–137. 2 cm
H	73.7–76.2 cm
I	40.6–43.2 cm

Tabel 1 - size and scale case studies (TU Delft, 2015)

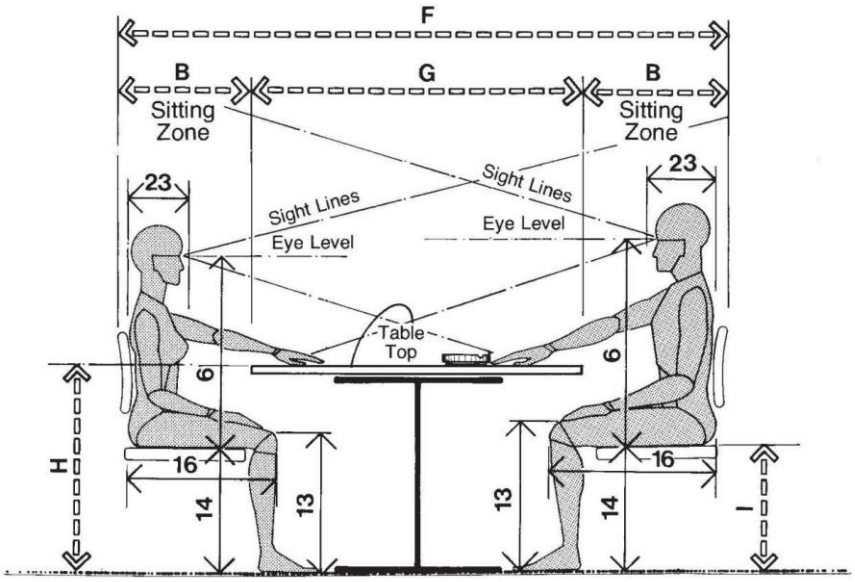


Figure 164 - Dimensioning of a conference table (Panero & Zelnik, 1979)

In the beginning, 3 cases have been described as possible use cases of the models. Alle these cases have an optimal scale, as discussed above. This scale translates to the required model size, this is the size the model at least needs, but more would be beneficial to get the area around a project. The scale and size can be seen in table 1.

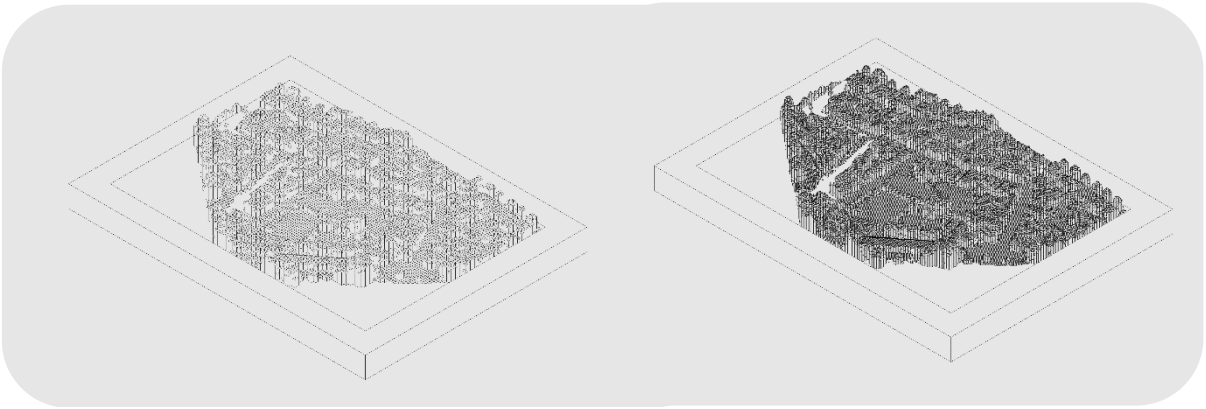


Figure25 - pin resolution A0 size, 10mm(left), 3mm(right)

Case studies				
nr	**description**	**Building site**	**Scale**	**Model size required**
1	dike house	15m x 26m	1:50	0,3m x 0,52m
2	parking lot city	40m x 80m	1:200-1:500	0,2m x 0,4m
3	Farming land	200m x 260m	1:500-1:1000	0,4m x 0,52m

Besides the model's scale and size, a particular resolution is needed. The resolution of the model is based on the pin size. A test has been done to see the difference in resolution, where a model is loaded on the format of an a0 paper, and then pin sizes of 10mm, 5mm, and 3mm are tested. The 10mm and 3mm results can be seen in figure 25. as seen in Figure 26, the resolution makes a big difference. Where the 10mm generates general shapes, buildings are not jet recognizable in that scale. The 3mm pins make buildings visible, and the roof structure can be seen. The visibility of building roof types, elevations, trees, and other obstructions makes this project different from most traditional and digital models like the 3D models on OpenStreetMap.

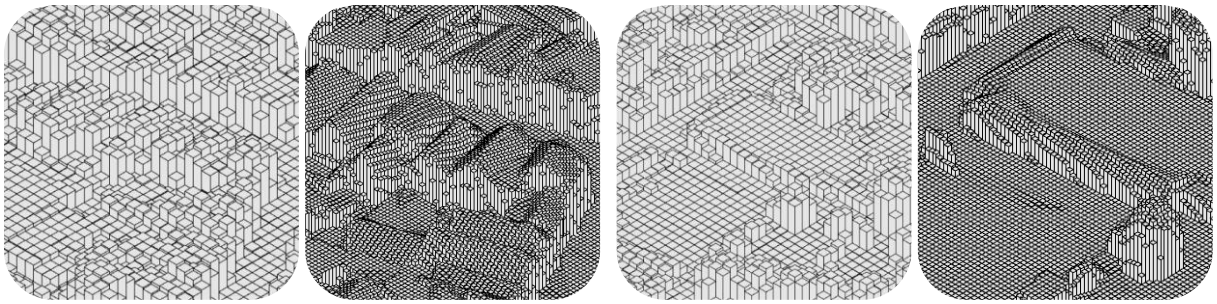


Figure 26 - matrix resolution 10mm and 3mm

Model size	scale	area	input	pins	Pin size	Comparison to pixels
1.2m x .84m	1:100	120mx84m	4pt / m2	40.320	5 mm	
1.2m x .84m	1:200	240mx164m	4pt / m2	157.440	2.5 mm	480P
1.2m x .84m	1:500	600mx420m	4pt / m2	~1m	2 mm	720P
1.2m x .84m	1:1000	1.2kmx.84km	4pt / m2	~4m	1 mm	1080p
1.2m x .84m	1:2000	2.4kmx1.6km	4pt / m2	~16m	.5 mm	4K

3. Machine Design

The following chapter is based on a design process. In this process, different options are tried to see what works. The process is not always based on the best solution but looks at a solution that works. The chapters are in order of progress during the time. Starting with the design requirements based on the outcome of the research and some practical requirements.

3.1 Design Requirements

Price, quality, and time are essential for the model and code requirements. Quality means the density and amount of pins inside the grid, also called the matrix. And time would be the time required to load the model.

general		
aesthetics	form follows function	
production Price	< €2000,-	
work surface	841mm x 1188mm	A0 paper size (chapter 2.5)
max outer dimensions	1400mm x 1800mm	
(Part) Weight	< 40kg	max liftable weight two pers
Expandable/adaptable		Adapt to different scales/use.
generation time	<1 hour	
Matrix		
Interchangeable	<1 hour	
Max deflection	1mm	
Max load	<1kg	
Matrix Density	no notable gaps	
Reset	<5min	
Pin holding	friction	
Manual pin movement		
Responsive feeling		
Possibility to show colors		
Pins		
Geometry	Square, uniform	
Components	<single component/pin	
Pin size	~2mm	
Pin work length	> 100 mm	
Resolution	~160.000	
Price	>€10.000/m3 material	
Wear	>1000 movements	
Material deviation	<5%	
Mechanism		
Precision	<0.1mm	
Movement speed	>100mm/s	
Safety:	Protected from user	
Energy consumption	non-when idle	
Assembly		
Assembly time:	< 8 hours	
Connections:	bolted/press fit	

3.2 Main design principle

Now that all the parameters have been set, the design of the machine can start. The basics are in the current machines, as described in Chapter 2. From this can be learned that individual controlling pins using a motor for every pin would not work. The best example could be seen in the discrete mould from the TU/e(seen in image 12). For this, the design can be separated into three parts. A layer with pins to move, a matrix layer, and finally, a layer responsible for moving pins up and down. We start with the moving mechanism, also described as the carriage.

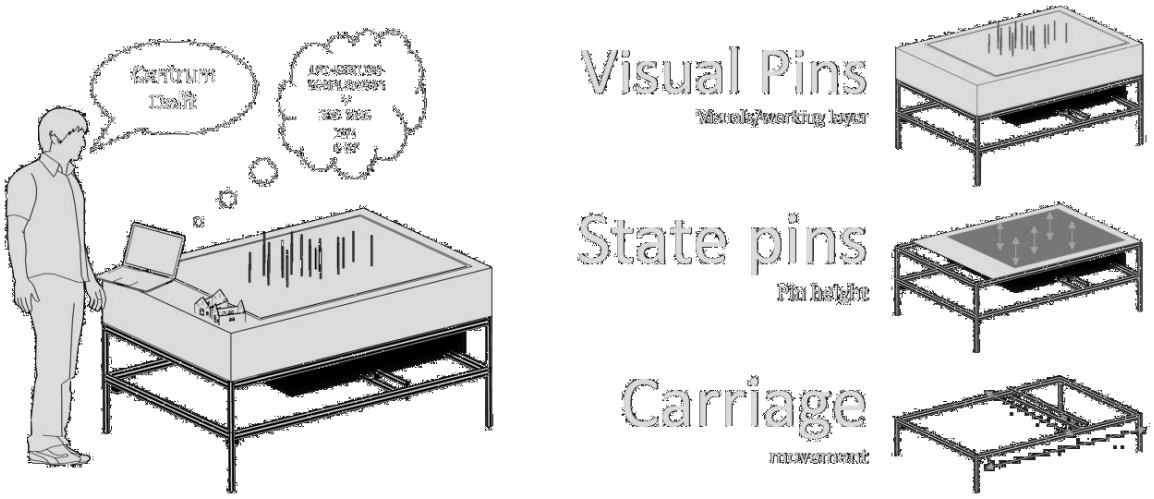


Figure 27 - general machine overview

3.3 Mechanism

For the mechanism, many options have been researched and tested. Looking at the current machines. Adhesive brakes, actuators, and motors would be excellent solutions. Due to the lack of knowledge about adhesive breaks, electronics, and magnetism, information about the electro adhesive breaks had to be gathered. Talking to experts in the field of electronics and magnetism, the general mention was that this would not be possible due to the power requirement for thousands of pins.

So the next step was solutions similar to DIY CNC machines are 3D printer systems. Much information about these topics can be found in the 3D printing handbook (Redwood, Schöffer, & Garret, 2017). Also, the website of Openbuilds (Openbuilds Design, 2023) can be handy in the design and software part of the moving mechanism.

The mechanism will be based on the fast 3D printer model called the CoreXY system, seen in figure 28. The main advantage of this movement type is the sturdy frame, rapid movement speeds, precision movement, power separation over two motors, and expandability, meeting all the requirements for the machine's structure, movement, and mechanism part.

Due to the cost limits, even the frame itself has been designed and manufactured myself. The main structure is based on the Openbuilds system with the design of an Ender 6 3D printer. A frame could be assembled with 3D printing parts and commonly used v-slot extrusion. Even the design of this frame and its components have seen many iterations.

The frame design started with defining the size. The main starting point was the Design requirements. Although these requirements are written for a full-size machine, the prototype could be much smaller. For this, the sizing is based on the available material.

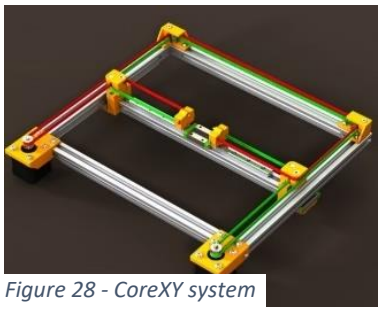


Figure 28 - CoreXY system

The next part of the design is the corner connection. At first, it had a dual function as a motor mount and pulley mount that would be used for the movement. After fiddling around with these brackets, the frame was not sturdy enough. There can be a separation between mounting the frame and the motors and pulleys. The corner bracket became the motor mount with integrated holes for the pulley system. With this, there were also some practical features to be added. How do you screw the bolt in, and how does it need to be manufactured? For this, additional holes were added for Allen keys and made simple so the parts could be 3D printed or laser cut and bent out of aluminum. The design process of this corner bracket can be seen in figure 29.

Figure 2917 – Design motor/pulley mount

On of the most significant differences in the sturdiness of the frame was the connection of the corners. In the first design, plastic brackets were printed that would slot over the corners and bolted to the frame with six bolts in every corner. As seen in figure 30(middle). Due to this connection, the frame was not sturdy enough for the carriage to smoothly ride the extrusions. In the second irritation, the extrusion's ends have been tapped with end holes drilled in the attached extrusions, as seen in Figure 30 (right). This method directly connects the extrusions, removing all the play in the frame.

Figure 3018 - Design corner connection, integrated with bracket (left), corner bracket (middle), tapped and bolted(right)

What makes this project different from other CNC machines is the z-axis movement. Due to the small size and fast-moving speeds, typical z-axis movements would not work. I looked into air, liquid, and mechanical solutions in the beginning stages. Air and liquid were quickly eliminated due to the need for additional components (compressors, tubing, etc.), noise, or cost. In the mechanical department, there are the options: stepper motors, pistons/actuators, and servomotors. In quick research,

pistons/actuators were insufficient for this project due to their limited extension or slow movement, which does not come close to the 100mm/sec as defined in the program of requirements. Stepper motors would be a possibility.

Due to the different possibilities for movement, the z-axis has seen many design changes. The main design principle was to keep the design of the gantry as close to the design of the side gantries as possible, requiring fewer unique parts. This meant that the z movement should be mounted on the gantry.

The first design was based on large stepper motors (Nema 17), like the driver motors on the corners for the x and y movement. This would be combined with a rock and pinion design to move a rod up and down, as shown in Figure 31. the plate spans way beyond the gantry and is very heavy on the outside due to the motors. The second design uses the same principle but with smaller motors(NEMA 8). With this size, four motors could be mounted to the gantry. This can be seen in Figure 31(middle). The next step was to ditch steppers and move to servos. A hobby servo motor is much smaller and cheaper than a nema8 motor. Another reason for the change is that the servos don't need a zero/end stop, removing the need for additional electronics on the moving part(z-gantry). A hobby servo can generally rotate 180 degrees. So in the 3rd design, a half gear was combined with a toothed rod. The problem with this method is that the height requirement for the movement is doubled, which means that the overall pin length is shortened. This can be seen in Figure 31(right).

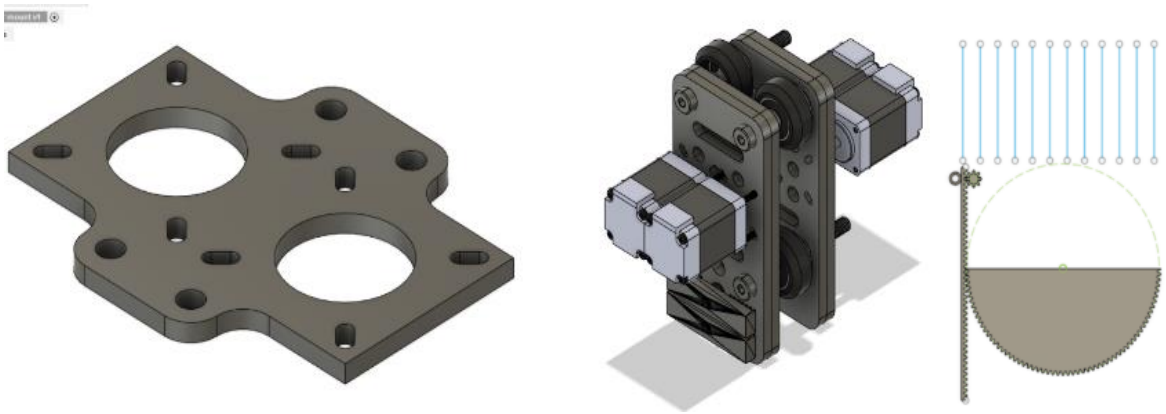


Figure 31 - z gantry itterations

The following design still uses the servos. However, the electronic servo is moved from the moving gantry to the electronics side of the machine, removing the need for power on the moving parts. In this design, the half gear can be placed on the side of the device, and with a Bowden cable, the movement is translated to the gantry. With this method, the gantry can be designed to be very small, and more servos could be added for a faster generation.

During the design, an estimate was made on the Bowden cable's radius. In this estimation, the Bowden cable would be 10x the tube diameter. As the pins are approximately 3mm, a Bowden cable of 3mm has been chosen. They resulted in a designed curve of 30mm after ordering to types of Bowden cables from Famotec. The radius was a lot bigger than expected. This can be seen in figure 32. The star-shaped design should have had much less resistance, but the radius is too big, as seen in the images. The other type can go much closer to the machine with its smaller radius but does not have the start shape design.

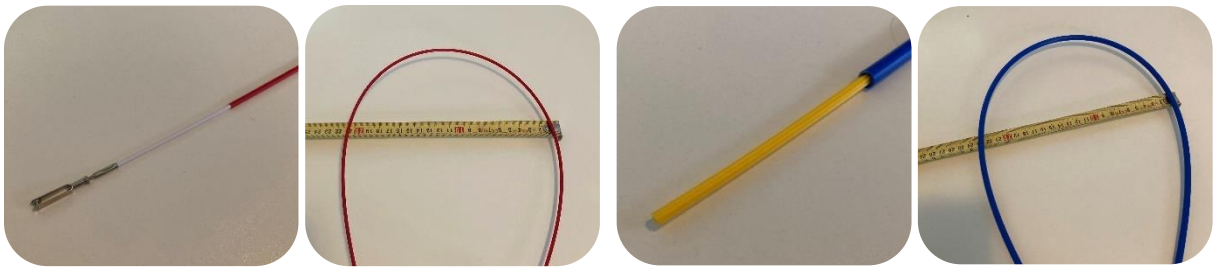


Figure 3219 - Bowden cables

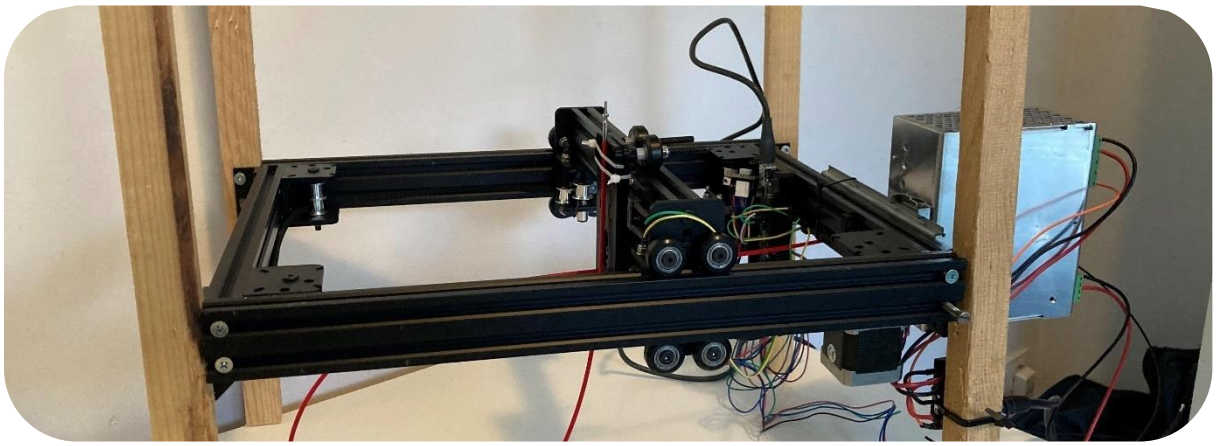


Figure 33 - prototype with Bowden cable

The motion to Bowden cable also has seen many iterations. Due to the stiffness of the Bowden cable, it was hard to design a small form factor driving mechanism. One of the first designs is shown in Figure 32(left). This design uses friction to push the inner cable into the Bowden cable. The plastic had no friction, so a rubber band was added. The gear was small, and the rubber band was continuously slipping off. So the second design, figure 32 (right), uses a larger rubber band, decreased tolerances, and springs were added to the underside of the Bowden cable to add pressure to the wheel. This fixed most of the problems. At his point, there was enough friction to push the Bowden cable. However, when the pins where added inside the matrix, more force was needed throughout the Bowden cable, and there was not enough pushing force. This lead to another design, where the Bowden cable was connected to the servo motor. Figure 32(right) in this design has also been chosen for another radius. As for the first design, only an extension of 4cm was manageable. The third design has an extension length of 24cm.

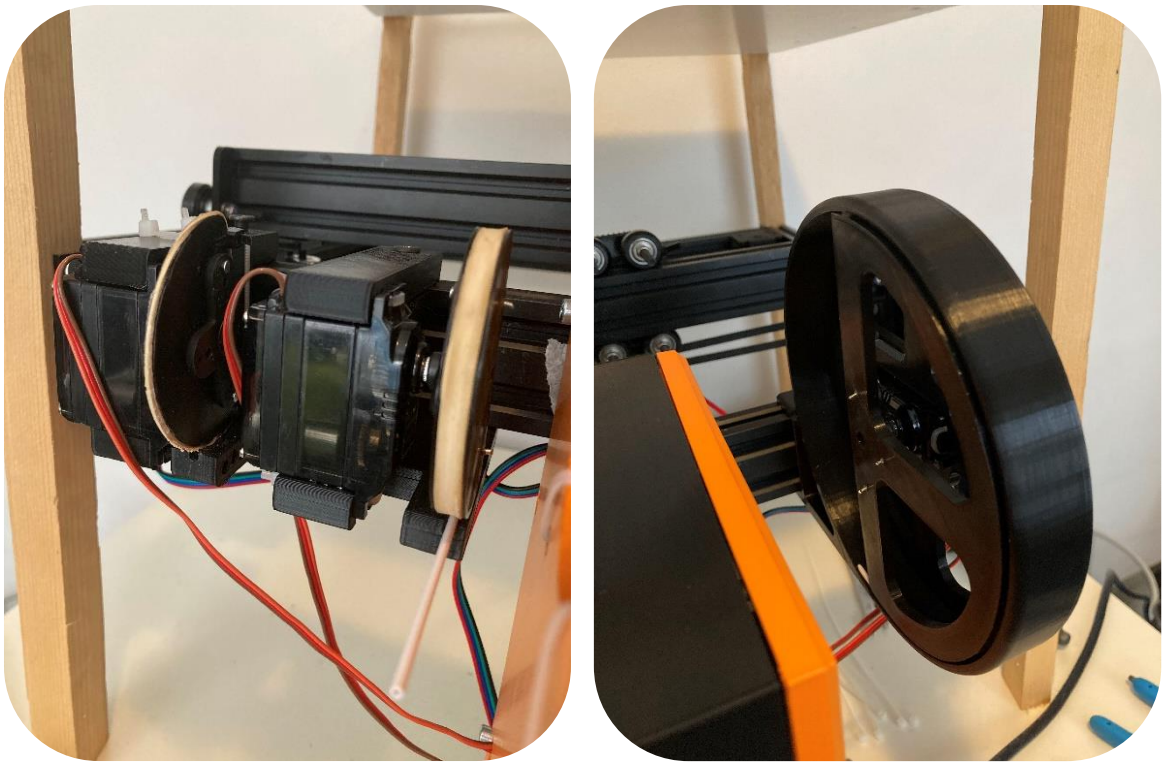


Figure 34 - motion to Bowden cable - design 1 and 2

Even though the second and third iterations have more friction, the machine could only move the Bowden cable. When a pin was inserted above the cable, the motor would stall. So the next step was to create a rigid connection to the motor arm. For this, the rack and pinion were used again. This can bee seen in figure 34

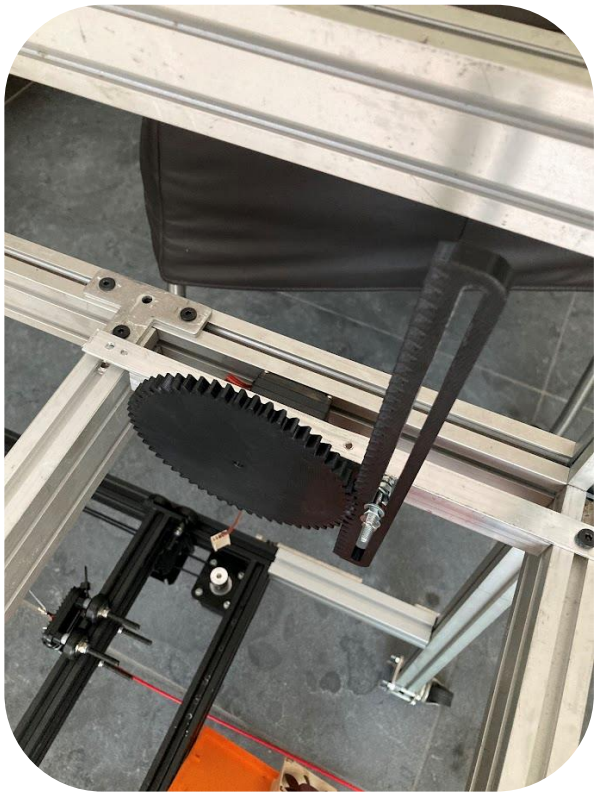


Figure 35 - rack and pinion design

The final workings of the mechanism can be seen in Figure 36. Here the z gantry can be seen with some of the Bowden cables. On the right side is the moving mechanism incorporated into the frame. The outer end will move up by moving the inner tube(white). Figure 37 displays this movement from the front. Moving the Bowden cable moves the metal rod up. This will, therefore, also move the visual pin-up. When the Bowden cable retracts, the metal rod will stay behind in the matrix/state layer(described in Chapter 3.5). When the Bowden cables are fully retracted, the z gantry will move over the x and y axis to the next position and repeat the steps.

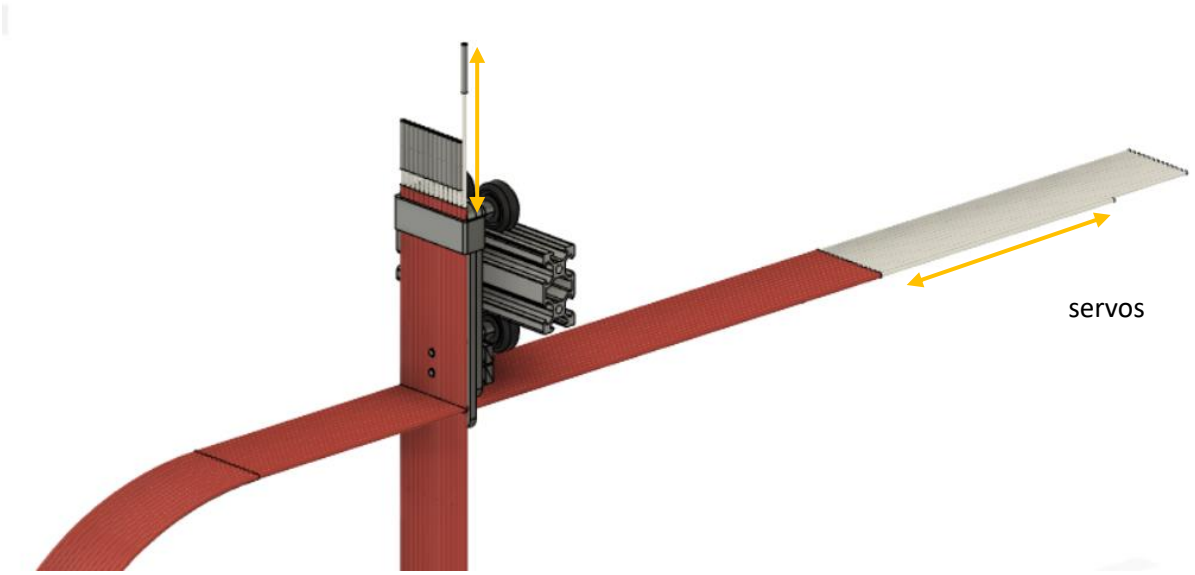


Figure 20 - z gantry with Bowden cables

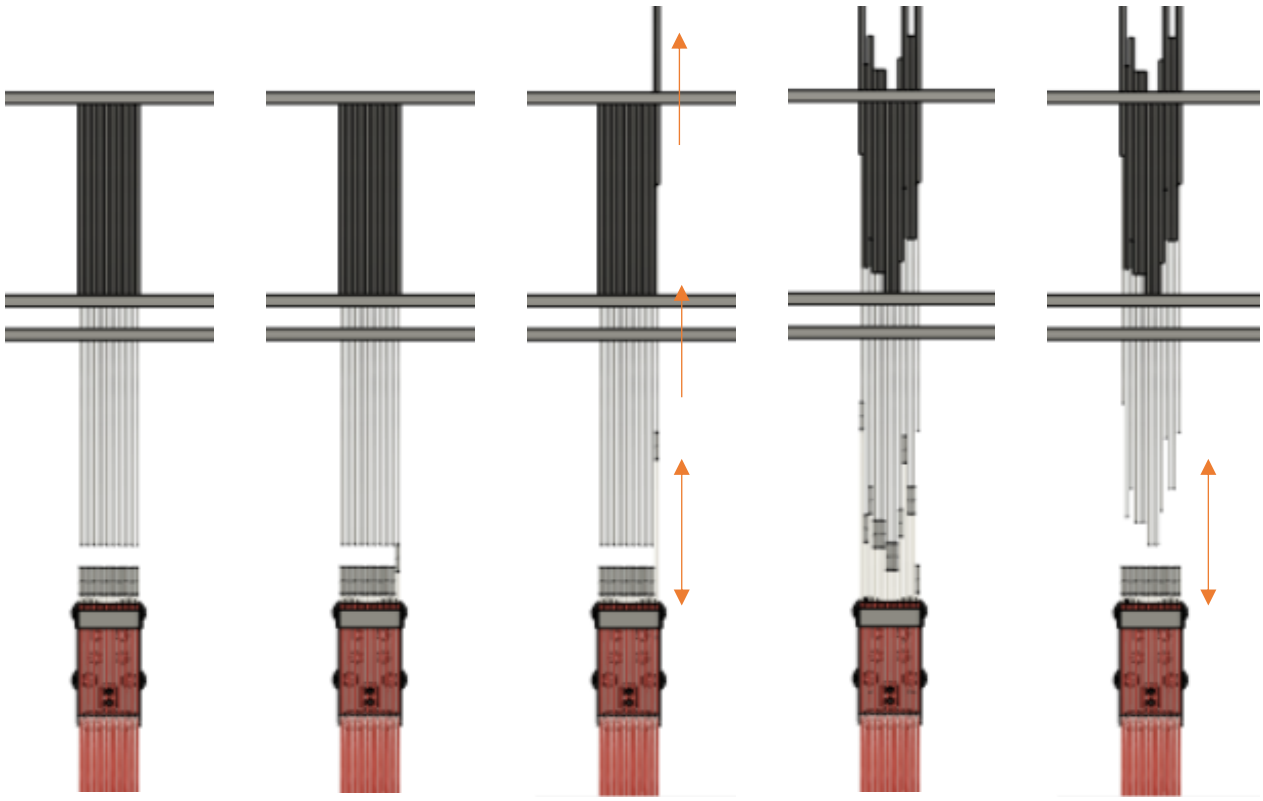


Figure 37 - pin movement

3.4 Electronics

For electronics, there are many options to use. Many boards are available for 3D printing and CNC cases, but more commonly used boards like an Arduino and computers like a raspberry pi can be used. Choosing the right type of board is research and can be different for any use case. For this research, five alternatives are looked at. The commonly used Arduino mega 2560 board is seen in figure 38. the second option is the same board but with the ramps v1.4 shield. And the other three options are from the DUET 3D lineup. (duet 2, duet 3 mini 5+, and the duet 3 6XD) These boards all have been chosen by personal experience.

Board 1 – Arduino mega 2560

This board is a commonly used board due to its affordability. It can be used by assigning individual pins. The board has 54 digital i/o pins, of which 15 can be used as PWM outputs and 16 analog input pins. The disadvantage of using an Arduino directly is that it is not explicitly made for controlling CNC machines and thus needs additional electronics like stepper drivers and power inputs. The board also does not have internal storage; therefore, a computer must be connected for the machine to work.

Board 2 – ramps v1.4

The ramps v1.6 is designed for 3d printing and CNC machines. The board is an add-on to the Arduino board. The difference with adding the shield is that the board already does a lot of power handling. With the 12v input, no additional stepper drivers are needed. This board also does not have internal storage.

Board 3 – DUET 2

The Duet 3D lineup is made explicitly for CNC machines. The board has integrated stepper drivers and can feed the stepper motors directly from power. The board has integrated storage and can be accessed via wifi or LAN.

Board 4 – DUET 3 Mini 5+

The mini 5+ is a newer version of the duet 2 and has more ports and higher feed power on a smaller board.

Board 5 – DUET 3 6HD

This board is the bigger brother of the mini 5. It is a bit larger but also has direct control for servo motors. All other boards do need a modification of the code or additional drivers. An overview of this board can be seen in figure 41.

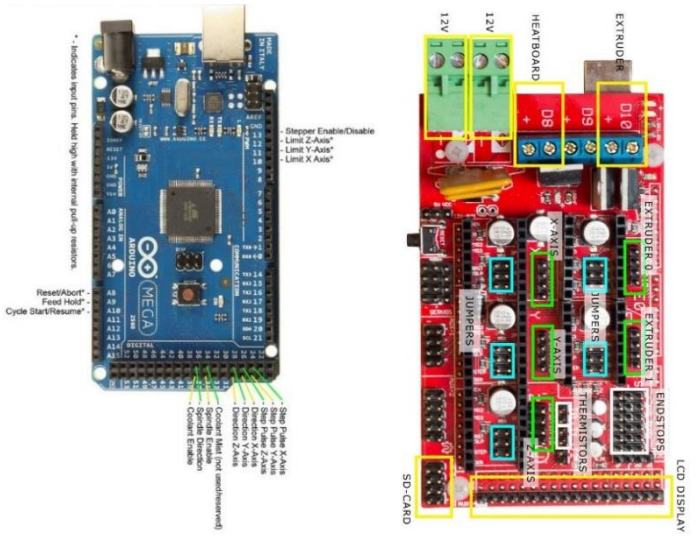


Figure 38 - wiring according to GRBL for Arduino mega (left) wiring for ramps (right)

Although the ramps v1.4 board is meant for 3D printers and CNC machines, it is not the best solution for this machine, as seen in figure 39. There are only four ports for servos. On the other side, a significant portion of the board is not used. For this, the second alternative was returning to the basic Arduino mega. With this, four digital ports are used for the stepper motors. This leaves 32 i/o ports for servos, as they only require one i/o port.

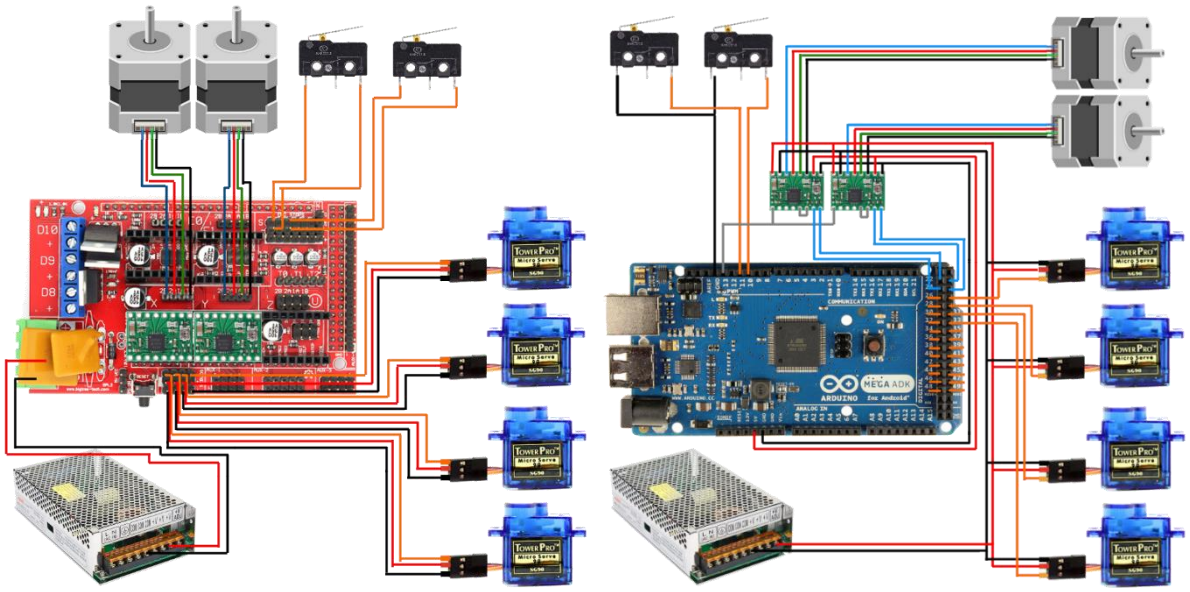


Figure 3921 - Ramps v1.4 Wiring diagram (left) Arduino mega wiring (right)

The Arduino mega with and without shield have been tested to get the model working. Software must be installed because the Arduino is not directly meant for CNC machines/3D printers. For this, GRBL has been used. GRBL Is an open-source, embedded, high-performance g-code-parser and CNC milling controller written in optimized C that will run on a straight Arduino. This g-code can be sent to the board, and the motors can be moved.

This combination of code and hardware(Arduino with GRBL) has seen much trouble during the research. The software did not always work as intended on the board, motors would not spin, and information was scares. After a week of trouble, the decision was made to swap all the electronics. To this point, available electronics had been used, not knowing if they would work properly. Besides, the Arduino mega was not an original board, which could also lead to some problems.

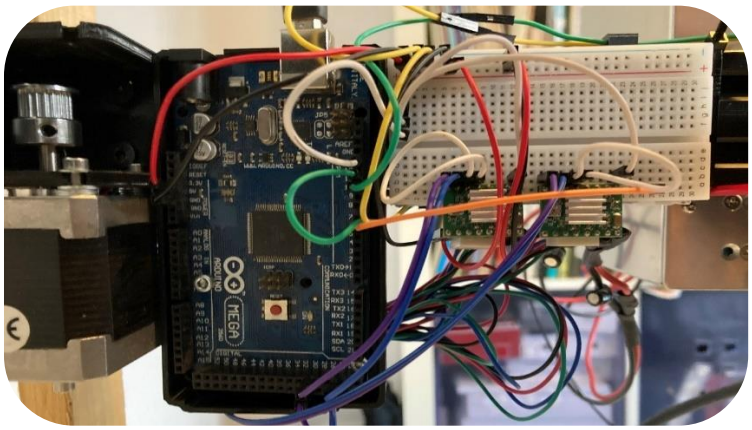


Figure 40 - Arduino connections

At this point, the decision was made to switch courses and go for the DUET 3D System completely. With this, all new components would be known to work, and the software would match the hardware. With duet, there are still four boards to choose from. To get the most adaptability for further research and the most servo connections, the DUET 3D 6HC mainboard has been chosen. This board has integrated stepper drivers and up to 10 servo connections using general IO pins. The integrated web control and storage also become very useful with these boards. This project could be stored on the machine and run via the web. This means there does not have to be a pc connected all the time, which was the case with the Arduino setup. An overview of the board and its connections can be seen in figure 41.

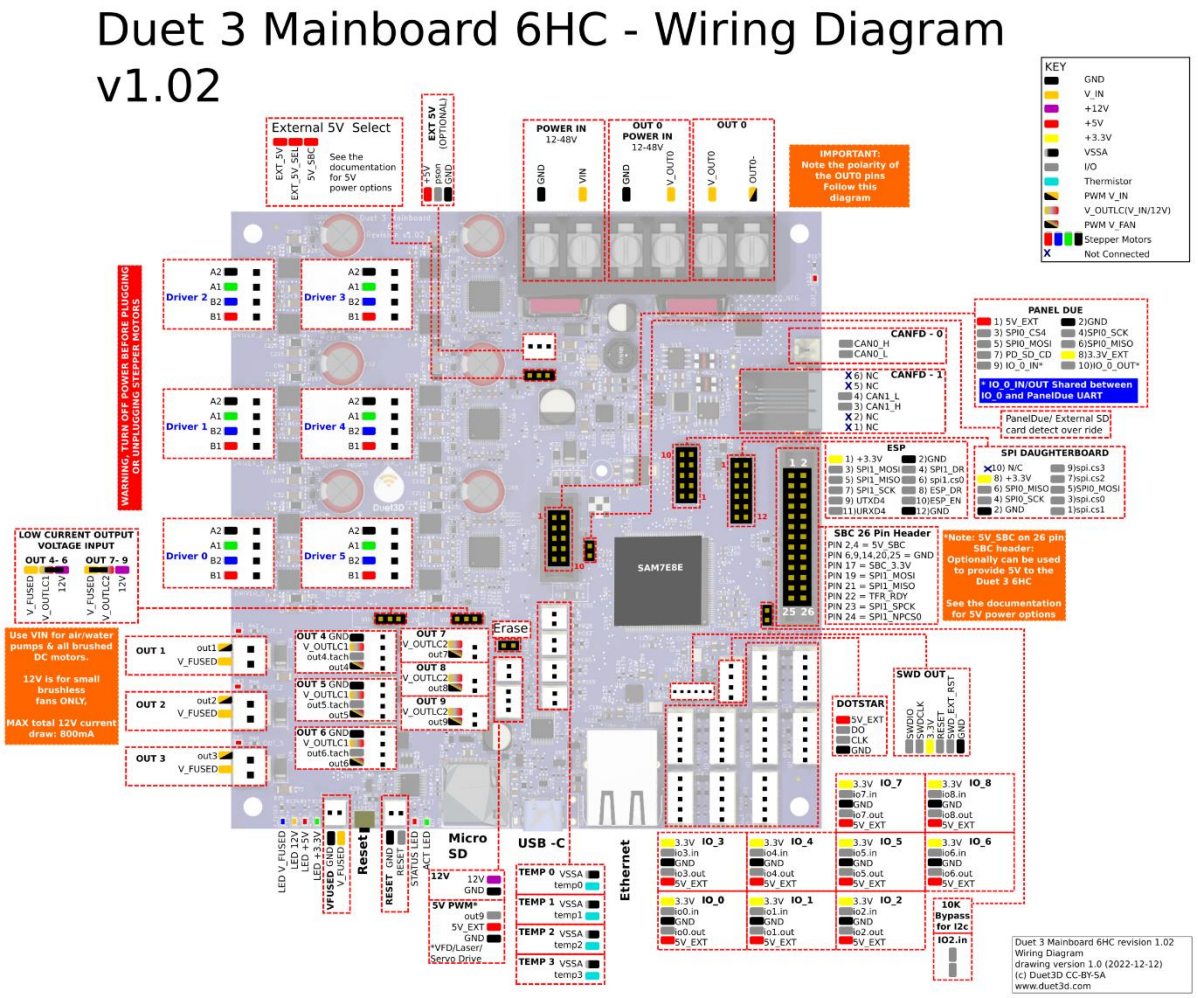


Figure 4122 - Duet 3 mainboard 6HC - wiring diagram

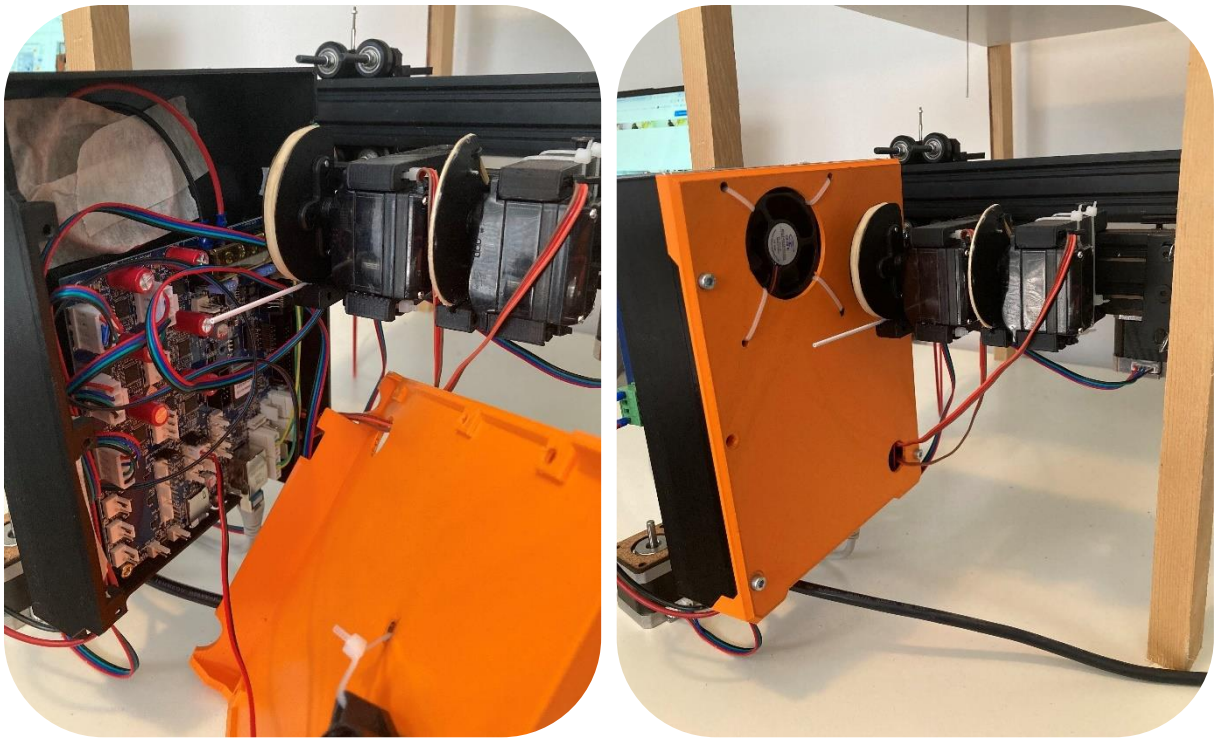


Figure 4223 - prototype electronics with duet board(left), enclosure(right)

With the new electronics from the duet, all settings can be changed using the web interface. These settings are well documented on https://docs.duet3d.com/ (Duet 3D, sd). In this research, the connectivity of a direct connection has been chosen, as seen in Figure 43. This is due to its use in different locations, like at home and school, making it unable to connect to one network. In another application, it would make more sense to connect the machine to your network so you can work on projects from all over the network. This application can be seen in the appendix under Manual Software.

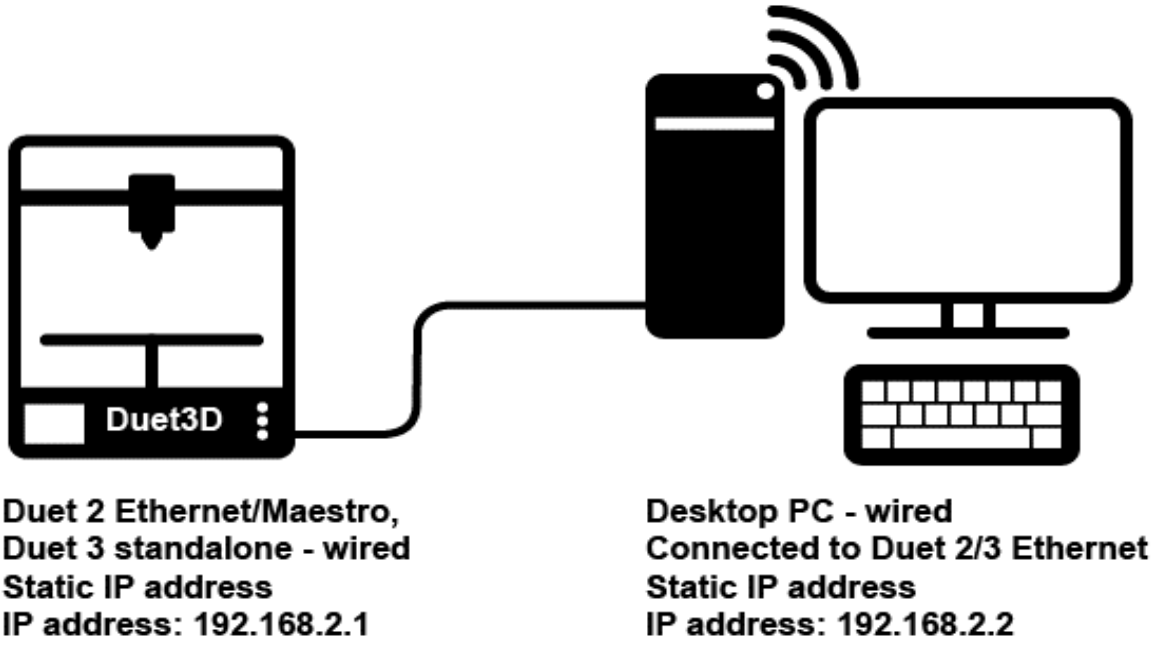


Figure 243 - duet connectivity setup

3.5 Matrix

The matrix is responsible for the pin's position. The function of the matrix is to maintain the pins in place during and after the movement. For this, a sturdy plate is needed that does not deflect due to the models' and pins' loads. To estimate this thickness, the following calculation is made for three common materials:

$$\Delta \max\ deflection(at\ centre) = \frac{0.0284wa^4}{Et^3(1.056\left(\frac{a}{b}\right)^5 + 1)} \qquad t = \sqrt[3]{\frac{0.0284w1400^4}{1.30056E}}$$

a = shortest span length (mm) = 1400mm
b = longest span length (mm) = 1800mm
w = load per unit area (MPa) = 9.8067 e^{-6}
E = Modulus of elasticity
t = thickness (mm)

Aluminum plate:
E = 69.000 MPa

$$t = \sqrt[3]{\frac{1.069.925}{89.700}} = 2.28mm$$

Plastics (POM):
E = 3.200 MPa

$$t = \sqrt[3]{\frac{1.069.925}{4.160}} = 6.36\,mm$$

Wood (multiplex):
E = 10.000 MPa

$$t = \sqrt[3]{\frac{1.069.925}{13.000}} = 4.35mm$$

With an "external" pin movement, the pins should be kept in place by another force than the driving motor/actuator used in most cases. Within the Discrete mould, tensioned dividers are used to generate a force to the side of the pins. In that case, a shifting pattern of pins is used, generating a curving divider that could easily be tensioned. This method would not work for this research as the grid distribution is unsuitable for curving dividers. The other problem is the need for tensioners adding more complexity and cost. The model described in this research also requires less force on the top, requiring less force to keep the pins in place.

Shifting plate Friction can be generated by placing a plate against the pins. With this, two plates will be used, one for holding the pins in place and one for generating force to the sides. This method comes with two problems. The first is the movement of the pins; with tolerances in the side of the pins, it would be possible that the pint would tilt due to the force applied to the side. This can be solved by adding a second shifting plate underneath the holding plate generating a balancing force. The second problem could also occur due to tolerances. When not all pins are the same size, more or less pressure would be applied to the pins.

Another option would be to add grippers to the pins. Adding small teeth to the pins with a rectangular shape would make it easier to move pins up and harder to push them down. The disadvantage of this process is that all the pins are custom-made, and fabrication contributes to this. With an ordinary 3D printer, pins can be printed to 5mm or smaller, and the principle of teeth would not work.

Another solution would be to remove the need to tool precise holes for the pins. By pressing the pins inside a material, all pin tolerances can be neglected as the material would fit exactly. Adding an elastic layer would even enhance as the material is pushed to the side instead of removed (what would be the case with tooling). In figure 44, a test can be seen with EPDM (Ethylene Propylene Dieen Monomeer). This material was ideal as it kept the pin in place. The next step was testing whether this material could hold its

Figure 44 - Pins in EPDM layer

grip power. For this, a test had been setup on the CNC machine, where a metal rod would go up and down 1000 times(based on design requirements). After this test, another pin was inserted next to the moved pin. Between these two pins, there was no notable loss of gripping power. Making this the ultimate solution for holding the state pins in place.

3.6 Visual pins

the visual pins are the most significant part of the machine as they will form the model. This means that the pins do not only have to meet the requirements but also look aesthetically pleasing. As the pin size determines the resolution, the pins must be as small as possible. Besides the size, there are also some other requirements the pins must check. Due to the fast amount of pins, it would not be affordable to make 100.000+ custom pins. To find the best materials for the application, Granta Edupack has been used. In the table below, the parameters can be seen. Figure 45 shows the outcome of this search.

Material finding 1 – Granta Edupack					
filter	material	price	density	hardness	>5wt% critical elements
value	Bulk materials/foams	€10.000/m3	2500kg/m3	0,1 HV	no

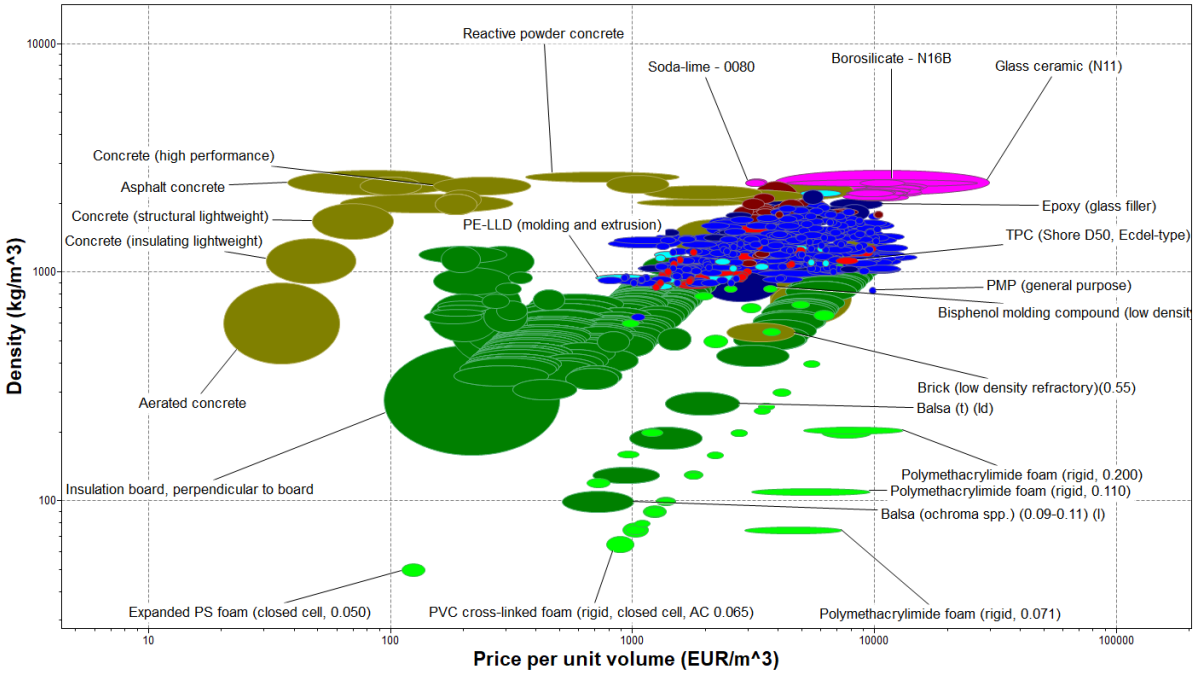


Figure 45 - material table based on Requirements

As seen in the image above, many materials would fit this search. Of the 4181 materials in the library, 1136 would pass the requirements. Upon further research, many of these materials would still not be

Figure 46 - material sizes and diviations

possible. For example, concretes and bricks cannot be produced in such small sizes and become brittle otherwise. The foams are cheap but can not withstand the force of a metal wire from below. Then there is the category of natural materials like wood. The balsa wood seems pretty promising initially, but after experimentation, the sticks can be very crooked, and the tolerances are relatively big. After the experiment, a second search was done. These parameters can bee seen in the table below.

Material finding 2 – Granta Edupack (narrowed sown search)				
filter	material	price	density	hardness
value	ceramics, composites, metals, and plastics	€ 40.000/m3	25.000 kg/m3	1 HV

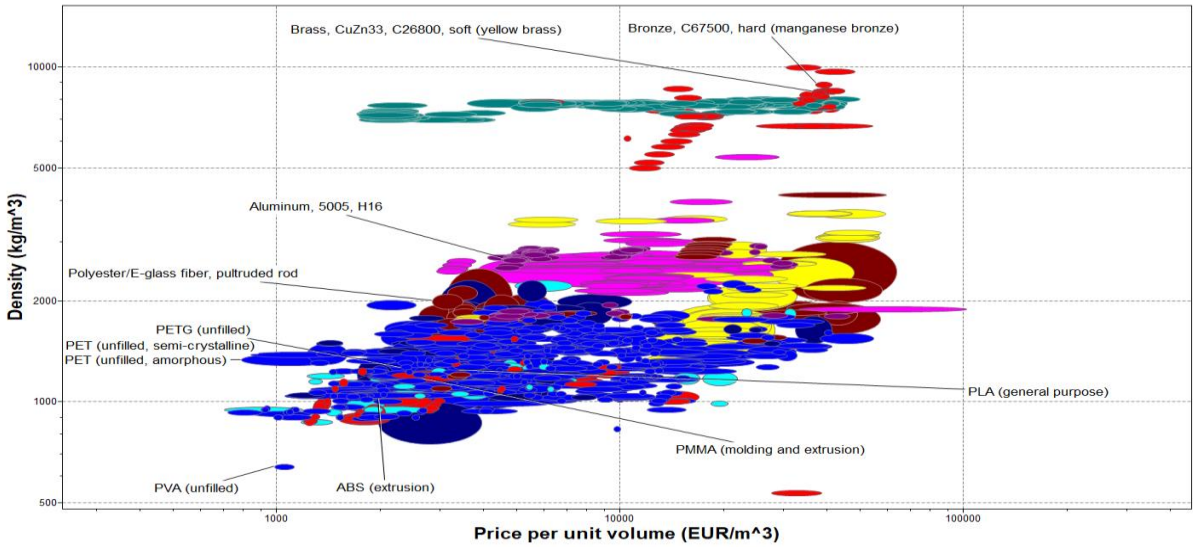


Figure 4725 - material table with changes

After closer inspection, only plastics would come close to the requirements, so the last refinement is to look closer into the plastics. Again the parameters and results are shown below.

Material finding 3 – Granta Edupack (narrowed sown search)					
filter	material	price	density	Additive	Filler
value	thermoplastics	€ 3.000/m3	1.400 kg/m3	None	None

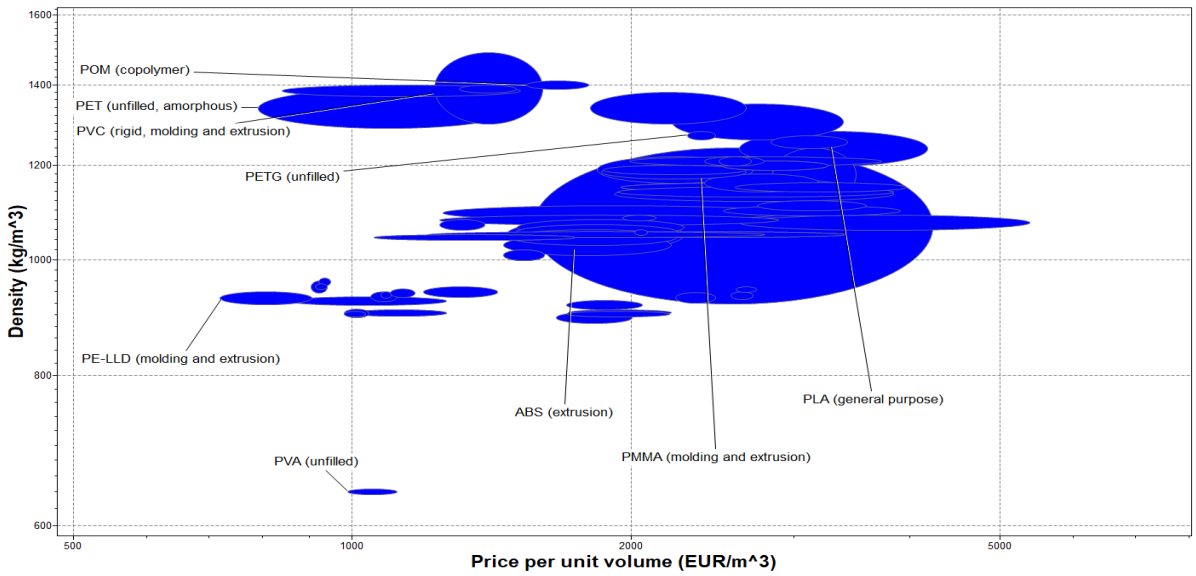


Figure 4826 - table of plastics meeting requirements

After the analysis in Edupack, different materials have been ordered for testing brass, aluminum, and plastics, consisting of PMME and PLA. The next step was to check for consistency in material size and the amount of deflection. Also, weight and cuttability have been studied, as seen in Figures 49,50, and 51. The materials where not the exact size as defined on the website. The materials that should be 3 mm actually where 3.1mm, and the one that should be 4mm is slightly smaller. Most delivery websites and manufacturers state that the material can be 5% to 10% bigger or smaller.

Another big thing to research is deflection. When the material extends to its max distance, it can be 200 longer, and this free standing from the other parts, the top should not deflect too much out as it will destroy the image. In Figures 50 and 51, the deflection can be seen over a span of 1m. The material is supported on both sides in the horizontal plane in the first image. The material is supported on the bottom and suspended in the vertical plane in the second image. From this, the metals deflect way less than the plastics.

The last part that has been checked is the weight. The weight can become huge when combining all the pins in a large-scale model. Even on the small-scale models, there is a considerable weight difference. This has some advantages as well as disadvantages. The disadvantage is that the weight will make the machine heavy, and a larger support plate is needed to keep deflection minimal. The

Figure 4927 - material deviations, brass (left), aluminium (middle), PMME (right)

advantage is that the pins have less impact on fractions. The gravity will pull the pins down, keeping them at the needed height.

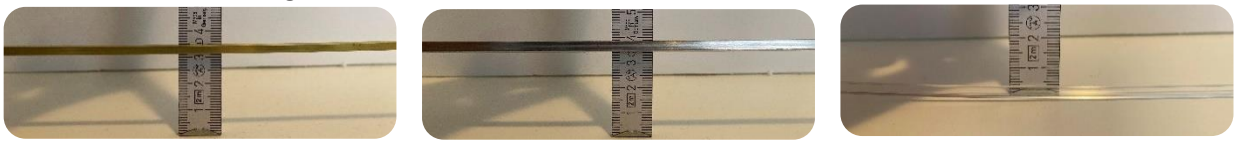


Figure 50 - material bending 1m span, brass (left), aluminium (middle), PMME (right)

Figure 5128 – material deflection 1m length, brass (left), aluminum (middle), PMME (right)

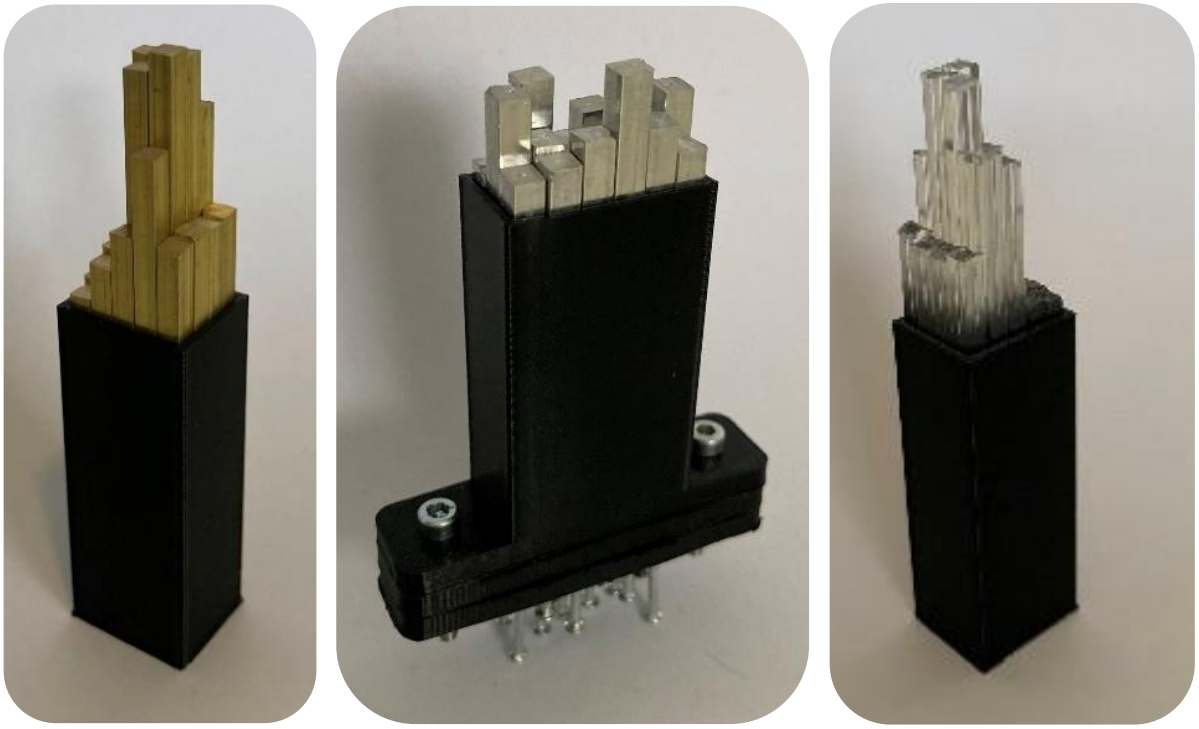


Figure 52 - pin appearance, brass (left), aluminum (middle), PMME (right)

On of the materials that have been tested is PMME, a translucent plastic material. One of the possibilities of the material is that it can let trough light. The light must be placed almost directly on the material, and the ends should be completely flat. As seen in Figure 52, it can be hard to see the differences due to the transparency when there is no light.

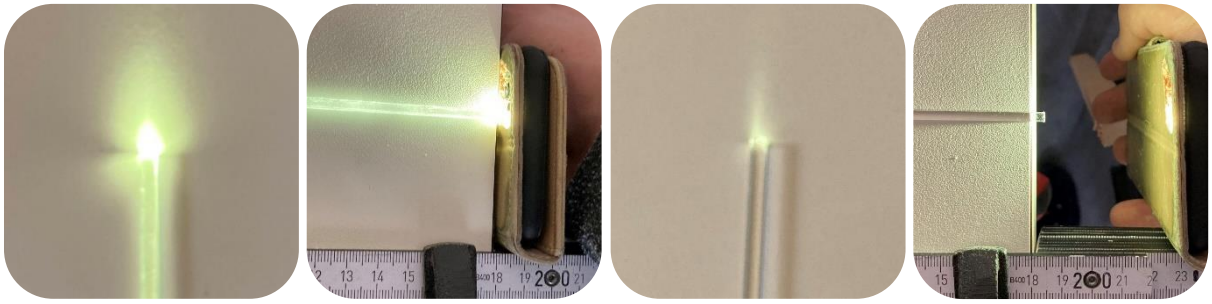


Figure 53 - light distribution PMME

In conclusion, after careful consideration, the brass pins emerged as the most aesthetically pleasing option. However, they were also the most expensive choice among all the available pins. After obtaining several quotations, I ultimately opted for HDPE (plastic) pins, which cost around 9 cents for an 18cm long pin, compared to one euro for the same size in brass. It's worth noting that the HDPE material was specifically designed for this application by tearing sheets into 3mm wide strips and cutting them to the desired length, while brass was delivered in 1m long straps intended for model making. The overall cost could be reduced by procuring the HDPE material in bulk and obtaining them in the required length directly from the manufacturer. The HDPE pins can be seen in the image below.

Figure 294- HDPE Pins, delivery(left), inside grid(right)

Aside from cost considerations, the size of the pins also played a significant role. The prototype's smallest possible pin size of 3x3mm was chosen to achieve a higher resolution in a smaller area. However, a pin size of 5x5mm would be more suitable in most cases. When generating an area using the code described in the subsequent chapters, a scale is selected for the application. With the 3x3mm pins, scale artifacts may become apparent when the scale is lower than 1:200, mainly when using the Tiff image method. However, this issue becomes less problematic with the pointcloud/laz method (slower). The 5x5mm pins would also be more appropriate for the image and interior methods, considering that most measurements involve increments of 100mm or sometimes 50mm. Furthermore, the scale used in these cases often corresponds to even numbers such as 1:20, 1:50, 1:100, or 1:200, making the 5x5mm pins the most suitable choice for larger scale tables.

3.7 Frame

To house all the components, a frame is needed. This frame could serve as a dual function, hold all the parts in place, and be a frame for a table. The first frame was made of v-slot profiles for the gantry system and wooden pillars as temporary feet. The matrix was made out of model-building foam. This method was advantageous in the early stages as it proved the system would work. However, it was not adaptable and would only work for the pin length at that moment(5cm). Besides that, it was not a very nice-looking frame. This frame can be seen in figure 55(left). The second frame was made out of a t-slot profile and made to table height. With these profiles and these heights, there was more room for changes. By sliding around the profiles, different lengths of pins could be tested. The frame was developed over time, and with the connections, frames could easily be slided around. The second frame can be seen in figure 55 (middle and left).

Figure 55 - frame development

For the possible costs and full-scale design, the frame builder of Ventio was used (Ventio, sd). With this designer, additional frames could easily be added, as well as all the connectors, feet, etc. The final design would cost around $1350. This design can be seen in figure 56.

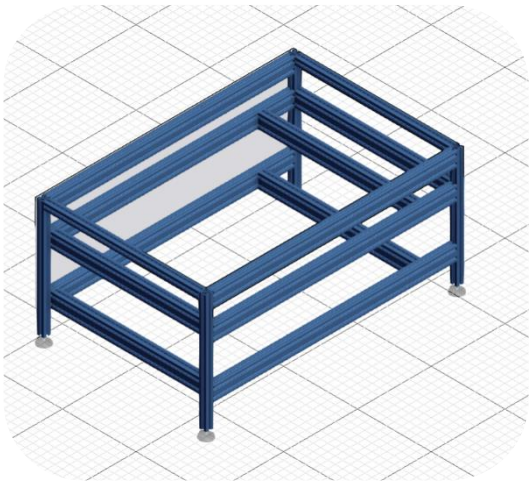


Figure 306 - frame design (ventio.io)

3.8 Assembling

With many screws, parts, and even more pins and rods to assemble, assembly is a big part of the research. With all the elements, assembling the machine has been in mind. The main part where assembly is of greet importance is with the pins. For the prototype, the pins had to be manually cut by a grinding wheel and trimmed with a belt sander. Doing this for all 350.000 or more parts would be very cumbersome. For this, standard solutions had to be thought out. Pins should be made out of standard material and not be connected with any thing else.

The other problem with the number of pins would be getting them into the grid. The holding/state rod could be placed by a machine as they are kept in place, but the pins themselves are another problem. Pins will likely go circular when arranging pins into a grid. Try to add a rubber band around a group of matches, and you will see. Tho overcome this problem, the machine can be tilted slightly to its side, and now a row can be placed on one side of the device. A thin piece of paper will sit between the layers for the next row. This paper will keep the layers separated and lower the possible friction of the pins.

3.9 Cost breakdown

Keeping the cost down was mandatory to make the machine an alternative to traditional methods. In this chapter, we will look at the expense of the prototype and what a full-scale model would cost. Please note that all these costs are consumer costs. In bulk orders, most of these products can become a lot cheaper.

	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
FRAME	V slot 2040 aluminium frame	6m	16m	€14,-/m	€134	€1350
					€134	€1350

	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
3d prints/aluminum parts	Motor mount	**4**		€5,-/pc	€10	
	pulley mount corner	**4**		€5,-/pc	€20	
	pulley mount gantry	**2**		€5,-/pc	€10	
	Gantry plate	**3**		€5,-/pc	€15	
	Z movement plate	**1**		€5,-/pc	€5	
	Electronics enclosure	**1**		€5,-/pc	€5	
	Limit switch mount	**2**		€1,-/pc	€2	
	Servo mount	**2**	**40**	€10,-/pc	€20	€400
	Cable binders	**4**	**8**	€0,25/pc	€1	€2
					€68	€259

	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
Screws and bolts	M3 – 10mm	8		€0,20/pc	€1,60	
	M5 – 12mm low profile bolt (mount to frame)	16		€0,20/pc	€3,20	
	M5 – 15 mm low profile bolt (mount to gantry)	8		€0,20/pc	€1,60	
	M5 – 20 mm low profile bolt (side gantry – y-axis)	4		€0,20/pc	€0,80	
	M5 – 40 mm low profile bolt (v wheels)	12		€0,20/pc	€2,40	
	M6 – 40mm hex bolt (frame)	16		€0,20/pc	€3,20	
	aluminium spacer m5 6mm	18		€0,05/pc	€0,90	
	Eccentric spacer m5	6		€0,50/pc	€3	
	M5 lock nut	28		€0,04/pc	€1,12	
	M5 washer	28		€0,01/pc	€0,28	
	unspecified				€18	
					€36	

Electronics	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
	24v power supply	1		€32,50/pc	€65,50	
	Duet 3D 6HC mainboard		1	€286/pc	€286	
	Nema 17 stepper motor	2		€15,50/pc	€31	
	Stepper cable	2		€1,70/pc	€3,60	
	Limit switch	2		€4,95/pc	€9,90	
	Hobby servo	4	4	€6,99/pc	€28	
					€424	

Electronics (optional)	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
	Duet 3HC	<4		€ 139,50/pc	€558	
	Servo	<32		€6,99/pc	€160	
	Nema 17 stepper motor	<2		€13,50/pc	€27	
	Stepper cable	<2		€1,70/pc	€3,40	
	Limit switch	<2		€4,95/pc	€9,90	
	LCD screen 7."	1		€ 114,50	€114,50	
					€872,80	

Pins	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
	Brass (not used)	260m	8.32km	€9,9/m	€2.574,00	€82.368,00
	Aluminum (not used)	260m	8.32km	€3,7/m	€977,60	€31.283,20
	HDPE	3500	112.000	€0.09/pc	€315,-	€10.000,-
	State pin	3500	112.000	€0.03/pc	€105,-	€3.360
					€420,-	€13.360,-

Other	Part name	Amount prototype	Amount Full machine	price	Total prototype	Total full machine
	V slot wheels	12		€2,75/pc	€33	
	Smooth pulley	**8**		€5,50/pc	€44	
	T5 pulley 16 teeth	**2**		€5,50/pc	€11	
	T5 belt	**2m**	**10m**	€4/m	€8	€40
	T5 belt clamp	**2**		€2/pc	€4	
	Bowden tubes	**2**	**40**	€3,80/pc	€7,60	€152
	Servo arm	**2**	**40**	€3,49/pc	€7	€140
	EPDM	**0,1m^2**	**2m^2**	€16,9/m^2	€1.69	€23,80
	Matrix layer	**1**		€	€	
					€120,30	€280,80

Total			prototype	Full model
	Frame		€134	€1350
	3D prints		€68	€259
	Screws and bolts		€36	
	Electronics		€424	
	Optional			€872,80
	Pins		€420	€13.360,-
	Other		€120,30	€280,80
	unaccounted	5%	€60	€800
		subtotal	€1.262,40	€16.922,60
	labor	8H*€60		€480
	profit	10-20%		€2.600
	tax	Neglectable due to the use of consumer prices		
				~€20.000

When comparing the prototype to the prototypes shown in the examples, this prototype has been a lot cheaper to make than, for example, the inform and the shapeshift. Where the prototype of inform costs 22500,- for 900 pins and Shapeshift 10.000 for 288 pins, this prototype could be produced much cheaper. As seen in the previous table, the device is only a fraction of the cost of the other machines on the market.

From a company standpoint, there are also some other factors to consider. As seen in the table above, there has been an approximation for labor and profit. This can still vary. The costs of development should also be taken into consideration. The table above only shows the cost of the physical machine. Besides that, there is also the cost of the software development. You can sell this separately to the machine as with other software products.

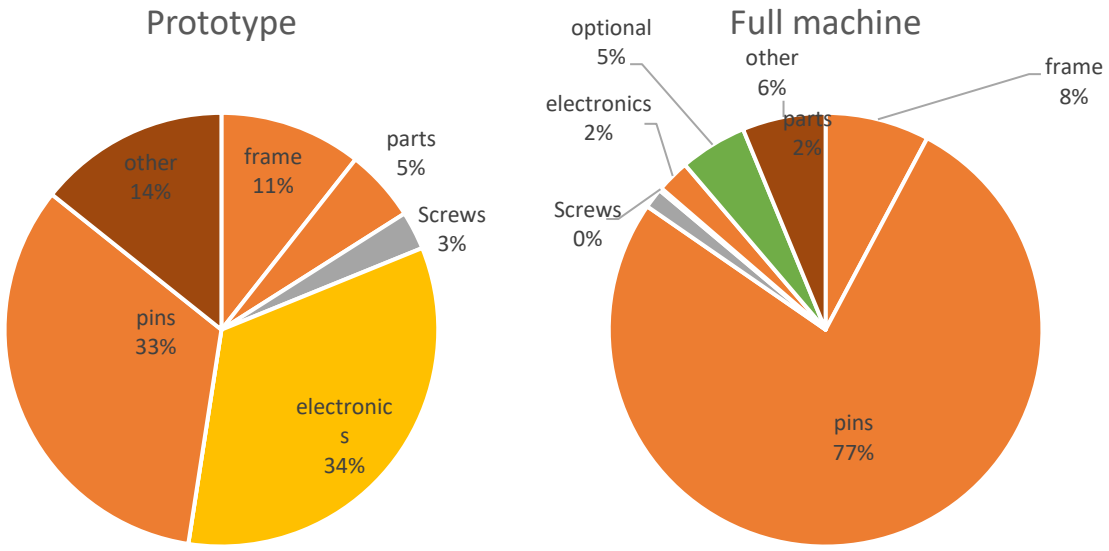


Figure 57 - cost breakdown, prototype(left), full machine (right)

4. Coding

A potential saving in time would be the addition of code to automate the process of generating the area. This chapter describes several ways to generate movements for the machine to work as fast as possible. During the research, different programs have been used for coding. In the early stages, rhino in combination with grasshopper was used. A visual representation could be seen to test if the code functioned adequately. The code has been rewritten in Phyton to make the project more standalone, as described in the program of requirements. A multi-use coding langue that is used in many fields. The coding of this project can be separated into multiple sections, starting with the graphical user interface (GUI) and going through all the options required for the program to work. A general overview of the program can be found in the appendix under the code overview.

4.1 GUI

For smooth use by the user, a graphical user interface(GUI) has been used and designed. As described in the design requirements, the software should be easy to understand and use. The GUI would consist of machine inputs, location/model input, and a visual representation. For this interface, we can import libraries into phyton. In this case, Tkinter is used. TKinter is a Python-integrated library for creating GUIs. Besides that, another library is used called CustomTkinter. This library is mainly used for visuals and some backend settings.

The GUI has seen some development throughout the research. Changes have been made based on additional content, user experience, or code optimization. The development can be seen below. The bases were made using an additional library called customtkinter. This can be seen in the first image. Then all the required parts were added, as seen in the second image. The final image is the latest GUI after iterations based on user input.

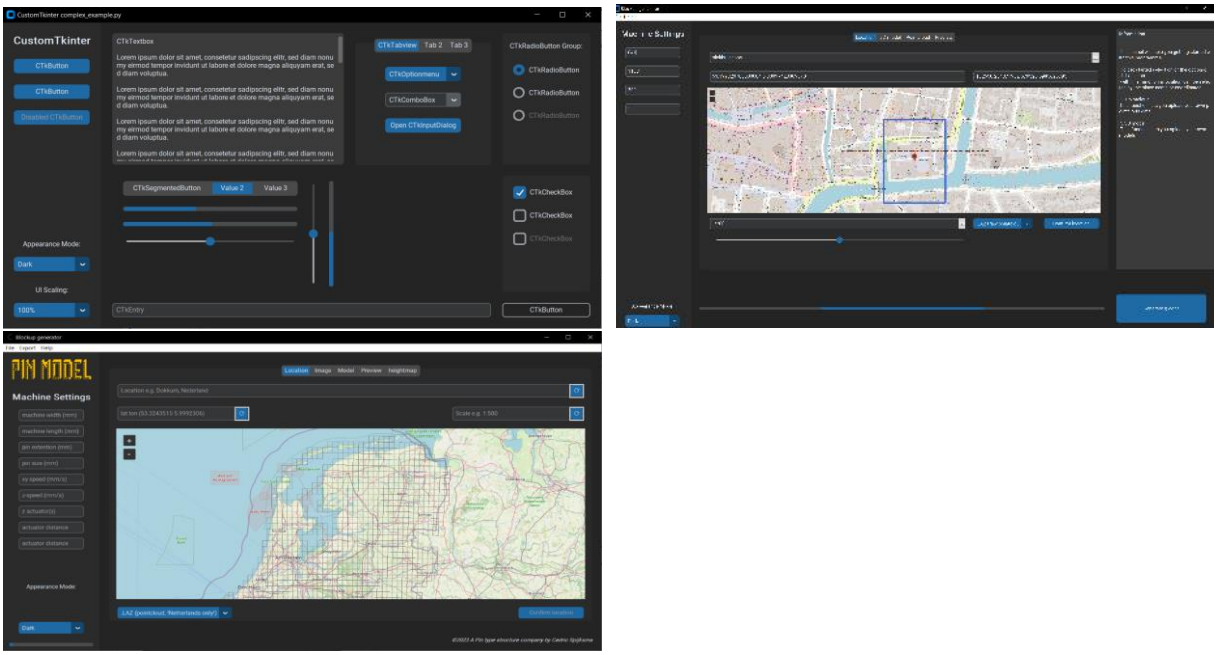


Figure 58 - GUI designs

As seen in the images, a map has been added for easy use. The library tkintermapview quickly added this map. This library will handle the display and the location conversion to coordinates, etc., as shown in Figure 58 (bottom left). Grid lines have been added to the map. This has been done to give a better

user experience on what maps are available and what maps have been downloaded. These maps will be explained later in this chapter.

In Figure 59, a part of the phyton code can be seen. This is the part where the map is added to the GUI. The first part is the setup of the “tab,” in this case, the “tab” location. The code will set up a grid comprising columns and rows. This will also define the row and column that would expand if you make the window bigger or smaller. Then in the contents part, the entries for the location, coordinates, and scale are added, and a dropdown menu for the map type. The for loop in the code will add refresh buttons and a “enter” command. These buttons and commands will help the code know when to proceed with the given information. The code's last part will add the map to the GUI. Additional lines were added to make it possible to change the map style and default starting location.

```
        # tabview 1 - location
        # setup layout
        self.tabview.tab("Location").grid_columnconfigure(0, weight=1)
        self.tabview.tab("Location").grid_columnconfigure((1,2), weight=0)
        self.tabview.tab("Location").grid_rowconfigure((0,2,3), weight=0)
        self.tabview.tab("Location").grid_rowconfigure(1, weight=1)
        # location tab content
        self.Location_adres = ctk.CTkEntry(self.tabview.tab("Location"),
                                           placeholder_text="Location e.g. Dokkum, Nederland")
        self.Location_adres.grid(row=0, column=0, columnspan=2, padx=(10,2), pady=10, sticky="nesw")
        self.Location_gps = ctk.CTkEntry(self.tabview.tab("Location"), width=250,
                                         placeholder_text="lat lon (53.3243515 5.9992306)")
        self.Location_gps.grid(row=0, column=2, padx=0, pady=10, sticky="nesw")
        self.Location_scale = ctk.CTkEntry(self.tabview.tab("Location"), width=250,
                                           placeholder_text="Scale e.g. 1:500")
        self.Location_scale.grid(row=0, column=3, padx=(2,10), pady=10, sticky="nesw")
        self.Location_type = ctk.CTkOptionMenu(self.tabview.tab("Location"),
                                               values=[settings.language.DSM,
                                                       settings.language.DTM,
                                                       settings.language.LAZ,
                                                       f'OSM (openstreetmap)'],
                                               command=self.update_grid_lines)
        self.Location_type.grid(row=2, columnspan=2, padx=10, pady=10, sticky="nsw")
        self.Location_confirm = ctk.CTkButton(self.tabview.tab("Location"), state="disabled",
                                              text=settings.language.confirm_location,
                                              command=self.confirm_location)
        self.Location_confirm.grid(row=2, column=3, padx=10, pady=10, sticky="nes")
        self.Location_text = ctk.CTkLabel(self.tabview.tab("Location"),
                                          font=ctk.CTkFont(size=12,slant='italic'),
                                          text=settings.language.right_click_help, anchor="w")
        self.Location_text.grid(row=3, columnspan=4, padx=10, pady=5, sticky="nesw")
        location_entries = [self.Location_adres, self.Location_gps, self.Location_scale]
        for i, entry in enumerate(location_entries): # creates a refresh end enter command for all location_entries
            entry.bind("<Return>", lambda e: self.get_coords())
            refresh = "refresh" + str(i)
            refresh = ctk.CTkButton(entry, text="⟳", width=4, command=self.get_coords)
            refresh.grid(row=0, column=1, padx=2, pady=2, sticky="NES")
        # location map tile
        self.Location_map = tkintermapview.TkinterMapView(self.tabview.tab("Location"), corner_radius=16)
        self.Location_map.grid(row=1, columnspan=4, sticky="nwse", padx=10, pady=10)
        self.Location_map.set_position(53.3243515, 5.9992306)
        self.Location_map.set_zoom(settings.default_zoom)
        self.Location_map.set_tile_server(settings.maptileserver)
        self.Location_map.add_right_click_menu_command(label=settings.language.add_marker,
                                                       command=self.add_marker_event,pass_coords=True)
        self.grid_lines()
```

Figure 5931 - Location code python

4.2 Machine input

For the machine to work correctly, inputs have to be given. The machine input is based on two parts, the parts you keep changing (user input in GUI) and the machine settings (based on the machine you have). The code must know all the machine limits for the machine to function correctly, including the device size, pin sizes, and length.

A settings file is created for these machine inputs you don't have to change frequently. This file stores information like the machine and pin size, movement speed, preferred language, and map style. Besides the settings file, you also have a sidebar in the GUI where you can easily change some settings.

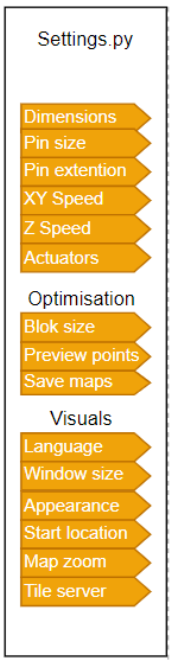


Figure 60 - overview settings

4.3 Mesh/Brep

One of the most common digital information is 3D models. 3D models are made out of points, lines, and planes. In programs, these models are also called Meshes or breps. As described in the introduction of this chapter, programs like Rhino and Grasshopper were used at first. The most accessible version for such a program was to create a model that would generate the pins out of a Mesh/Brep. The workflow for a mesh/Brep to g-code is as follows:

1. Generate a grid of points based on dimensions and pin size
2. Make a line out of the point with the max pin length
3. Load mesh/brep
4. Check for crossing between mesh/brep and lines and generate point
    a. Take these points and extract z value
    b. Lower the whole model based on the lowest z value
    c. Lower all z values above the max pin length to pin length
    (These values can be used for the g-code)

A general overview of this method can be seen in the image/workflow below

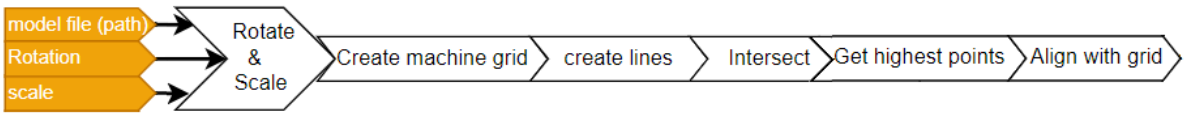


Figure 61 - workflow code mesh/Brep/3D model

Within Grasshopper, this method was quickly done. In Phyton, however, this gave many problems. 3D modeling programs have integrated functions for loading models, and finding intersections proved to be a lot harder in Phyton. Combine this with around the generation of 300.000 points for a full-scale model, and you have many calculations to make.

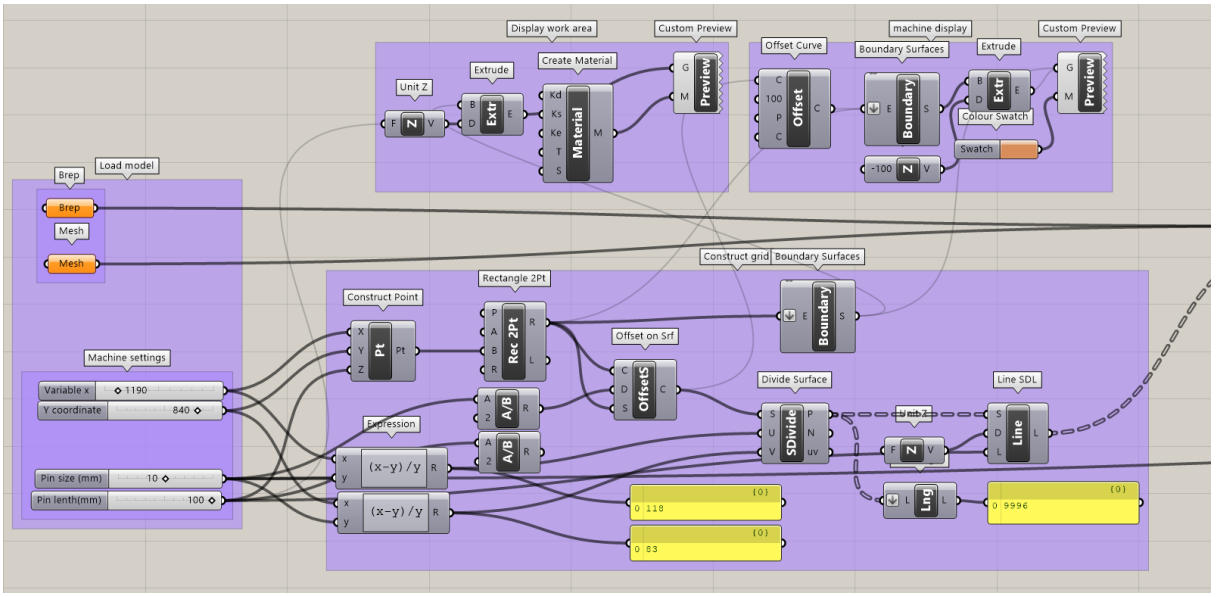


Figure 62 - mesh/brep in grasshopper

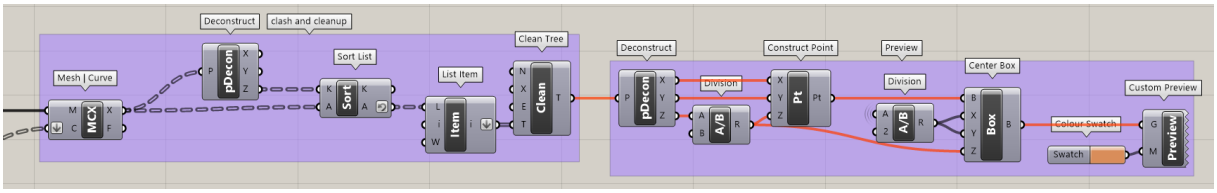


Figure 63 - height generation grasshopper

4.4 Pointcloud

The point cloud has another working method, with no closed surface to clip points from. For this, several ways are possible. Creating a mesh and running the previous script would be a possibility. Jet software cannot do this reliably and requires much computational power. A workaround would be to generate planes on the position of all the points and clip with the lines in the previous script. This also is computationally very heavy as thousands of lines are compared with up to millions of planes. This could be solved with the use of the hyper-plane method. In this method, coordinates are divided into several groups. And then compare the groups instead of the whole sets. Still very computationally demanding. The cloud could be randomly reduced based on its size. Reducing the computational power required. But the least demanding method that eventually was used is as follows:

1. Load point cloud
2. Random reduce point to (2x the number of pins)
3. Moving the cloud to the workplane
4. Reduce the z value of all high points to max height
5. Clip outside points
6. Deconstruct points into individual components
7. Divide by pin size, decimate, multiply by pin size, move -½ of pin size
   (now the coordinates are reset to the grid points)
8. Remove duplicates
9. Filter based on z value
10. Get the highest point/coordinate
11. All that remains is the highest point locations that can be used for g-code

12. For representation, see 5. From the previous workflow

This can also be seen in the workflow below, made in Grasshopper.

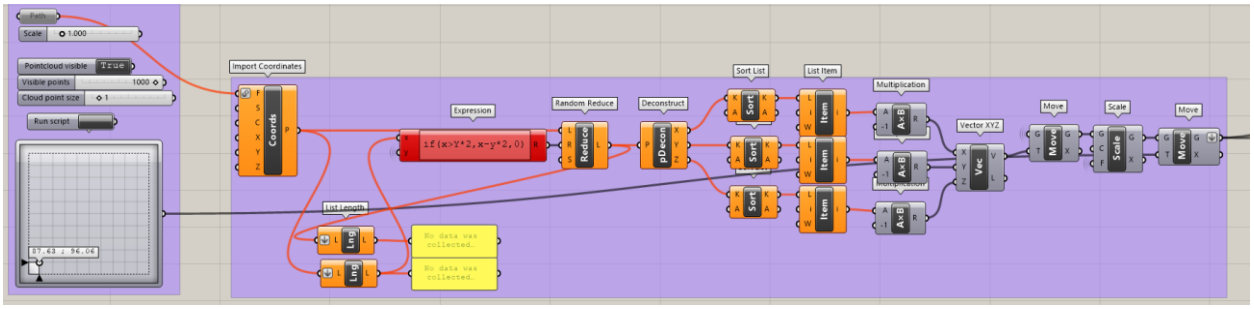


Figure 64 - pointcloud conversion grashopper

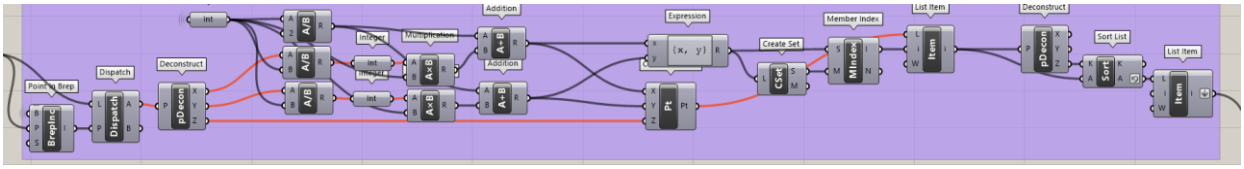


Figure 65 - point cloud cleanup grasshopper

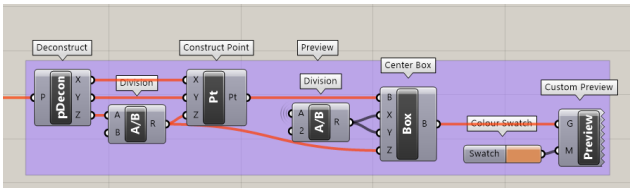


Figure 66 - preview grasshopper

In phyton, it would look a bit different. This can be seen in Figure 67.

```
149 def las_2_points(kaartbladen,rd4,rd2,pinsize,scale,pin_extention):
150     print("start cropping")
151     laz_files = [os.path.join('TEMP/C_', "C_" + str(k.upper()) + ".LAZ") for k in kaartbladen]
152     x_max = rd2[0]
153     y_max = rd2[1]
154     x_min = rd4[0]
155     y_min = rd4[1]  # based on rd coordinates
156 
157     all_points = [] # Initialize an empty list to store the cropped points from each LAS file
158     for laz_file in laz_files:
159         print(f"reading {laz_file}")
160         las = laspy.read(laz_file)
161         mask = (las.x >= x_min) & (las.x <= x_max) & (las.y >= y_min) & (las.y <= y_max) # Crop the points on the scaled var
162         points = np.vstack((las.x[mask], las.y[mask], las.z[mask])).T # Combine the x, y, and z points into a numpy array ar
163         all_points.append(points)
164         print(f"{laz_file} cropped and combined")
165 
166     all_points = np.concatenate(all_points, axis=0).astype(np.int32) # Concatenate all the points into a single numpy array
167     num_points = 800000  # Replace with 4x/8x the amount of pins
168     if len(all_points) > num_points:
169         idx = np.random.choice(len(all_points), num_points, replace=False)
170         all_points = all_points[idx]
171     all_points = all_points * 1000 / scale # scale the points
172     all_points = np.round(all_points, decimals=3)
173     points = all_points - np.min(all_points, axis=0) # move the points to start at (0, 0, 0)
174     print("files cropped and combined")
175     toppoints = Points_2_toppoints(points,pinsize,pin_extention)
176     return toppoints
```

Figure 6732 - pointcloud conversion phyton

```python
def tif_2_points(kaartbladen, rd4, rd2 ,pinsize,scale,pin_extention, filetype):
    all_points = [] # Initialize an empty list to store the cropped points
    if filetype == "dsm":
        filetypetext = "r_"
        directory = "TEMP/R_/"
    elif filetype == "dtm":
        filetypetext = "m_"
        directory = "TEMP/R_/"
    for kaartblad in kaartbladen:
        print(f"reading {kaartblad}")
        kaartbladen = pandas.read_excel('kaartbladen.xlsx')
        row = kaartbladen.loc[kaartbladen['bladnr'] == kaartblad]
        lo_x, lo_y = row['lo_x'].values[0], row['lo_y'].values[0]
        dsm_file = directory + filetypetext + kaartblad +'.tif' # Define the file path to the DSM file
        # Define the coordinates of the bounding box to crop
        left = int(max(rd4[0] - lo_x, 0)*2)
        right = int(max(rd2[0] - lo_x, 0)*2)
#         bottom = int(max(rd4[1] - lo_y, 0)*2)
#         top = int(max(rd2[1] - lo_y, 0)*2)
        bottom = int(max(12500 - (rd4[1] - lo_y)*2, 0))
        top = int(max(12500 - (rd2[1] - lo_y)*2, 0))

        with rasterio.open(dsm_file) as src:
            win = Window.from_slices((top, bottom), (left, right)) # Create a window from the pixel coordinates
            dsm = src.read(1, window=win) # Read the DSM data within the window
            transform = src.transform # Get the georeferencing transform

        # Convert the DSM to x, y, z coordinates
        rows, cols = dsm.shape
        x,y,z = [],[],[]
        for row in range(rows):
            for col in range(cols):
                px, py = rasterio.transform.xy(transform, row+win.row_off, col+win.col_off)
                x.append(px)
                y.append(py)
                z.append(dsm[row, col])
        all_points.append(np.vstack((x, y, z)).T) # Combine the x, y, z coordinates into a list of points
        print(f"{kaartblad} cropped and combined")

    all_points = np.concatenate(all_points, axis=0).astype(np.int32) # Concatenate all the points into a single numpy array
    all_points = all_points * 1000 / scale # scale the points
    all_points = np.round(all_points, decimals=3)
    points = all_points - np.min(all_points, axis=0) # move the points to start at (0, 0, 0)
    print("files cropped and combined")
    print(len(points))
    toppoints = Points_2_toppoints(points,pinsize,pin_extention)
    return toppoints
```

Figure 68 - point cloud conversion using Tiff

4.5 Image conversion

During the project, more and more options were found to be used in the architectural and engineering context. One of these options is to convert images. By converting images, interior designs can be made by giving the machine the wall's location or representing data in a 3d graph. These images can be data like population density or energy usage etc. The workflow for this coding is entirely different from the other two solutions the workflow:

1. upload image
2. crop image to the size
3. rotate/invert the image
4. convert image to grayscale
5. extract pixels and their grayscale
6. interpolate the grayscale over the length of the pins
7. convert the pixels and values to their coordinates used for g-code
8. present the preview like the other options.

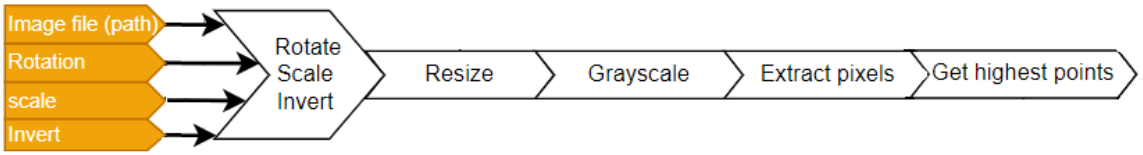


Figure 69 - workflow code point cloud

```python
    def confirm_image(self):
        self.progressbar.set(0.2)
        self.values()
        width = self.machine_width/self.pin_size
        length = self.machine_length/self.pin_size
        img = Image.open(self.openImage_path, "r")
        img = img.rotate((int(self.rotate_image.get() or 0)*90))
        if self.image_scale.get() == 1:
            img = img.resize((width, length))
        else:
            print("not implemented yet")
            messagebox.showerror("Error", f"Not implemented yet, using default variant")
            img = img.resize((width, length)) #temporary
        img = img.convert("L")
        img = ImageOps.mirror(img)
        if self.image_invert.get == 1:
            z = (((numpy.asarray(img))/255) * self.pin_extension).astype(int)
        else:
            z = (((255-numpy.asarray(img))/255) * self.pin_extension).astype(int)
        rows, cols = z.shape
        x, y = numpy.meshgrid(numpy.arange(cols), numpy.arange(rows))
        x, y = x*(self.pin_size), y*(self.pin_size)
        coords = numpy.column_stack((x.ravel(), y.ravel(), z.ravel()))
        toppoints = coords[coords[:, 2] != 0, :]
        self.toppoints = toppoints
        self.preview_show(self.toppoints)
```

Figure 70 - image conversion phyton

4.6 Point Cloud download

Data collection is one of this project's main advantages compared to other working methods. Much time can be saved by removing the need to convert files and information. One option to collect the data based on location is to use the point cloud data available through the Ahn in the Netherlands. This option thus only works in the Netherlands, and other solutions should be looked at for other countries. Due to time constraints, this is the only available option for now. The workflow for collecting the point cloud data from Ahn is as follows:

1. extract coordinates of a location
2. convert WPS (GPS coordinates) to the Dutch Rd. coordinate system
3. lookup the coordinates in a lookup table and find the required kaartblad(map on which the coordinates are)
4. check if all the data is on the same map or if additional maps are needed
5. create a downloading URL using the Kaartblad
6. download the data and place it in the TEMP directory
7. from here on, the data can be loaded like the point clouds described above.

This method of converting point clouds works and is precise, but it can take 30-40 minutes for the whole process to work. This led to the changing of the method. Pdock, the supplier of the point cloud, also delivers tiff files with a bit less precision (4 points/m2 instead of 8-15/m2) but is in a more manageable file format. With this tiff file, the download time has gone from 15 minutes to 30 seconds. The cropping can be done more efficiently as with a tiff file. Everything is stored in rows and columns, cropping quickly to those rows and columns, reducing the time even further. With this new method generating times of 2 minutes can be achieved.

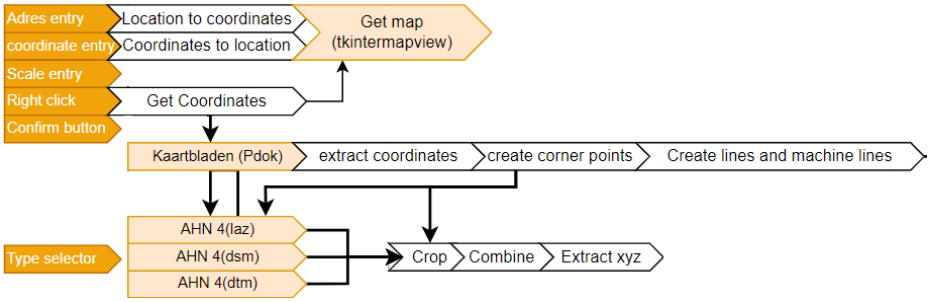


Figure 71 - workflow code downloading point cloud data

```python
def find_kaartblad(rd1,rd2,rd3,rd4):
    def find_map(point):
        kaartbladen = pandas.read_excel('kaartbladen.xlsx') # Load data into a dataframe
        closest_distance = float('inf') # Initialize a variable to store the closest distance and the corresponding bladnr
        closest_bladnr = None
        for index, row in kaartbladen.iterrows(): # Iterate through the rows of your dataframe
            x, y , bladnr = row["lo_x"]+2500, row["lo_y"]+3125, row["bladnr"]
            distance = math.sqrt((point[0] - x)**2 + (point[1] - y)**2)
            if distance < closest_distance: # Check if this distance is smaller than the closest distance so far
                closest_distance = distance
                closest_bladnr = bladnr
        return closest_bladnr
    kaartblad1,kaartblad2,kaartblad3,kaartblad4 = find_map(rd1),find_map(rd2),find_map(rd3),find_map(rd4)
    kaartbladen = [kaartblad1, kaartblad2, kaartblad3, kaartblad4]
    unique_kaartbladen = list(set(kaartbladen))
    print(unique_kaartbladen)
    return unique_kaartbladen
```

Figure 72 - kaartblad selection phyton

```python
def download_map(kaartbladen, filetype):
    os.makedirs('TEMP', exist_ok=True)
    for k in kaartbladen:
        if filetype == "laz":
            os.makedirs('TEMP/C_', exist_ok=True)
            url = "https://ns_hwh.fundaments.nl/hwh-ahn/ahn4/01_LAZ/C_" + str(k.upper()) + ".LAZ"
            file_path =  os.path.join('TEMP/C_',"C_"+str(k.upper())+".LAZ")
            if not os.path.exists(file_path):
                print("downloading " + url)
                r = requests.get(url, stream=True)
                with open(file_path, "wb") as f:
                    f.write(r.content) # save file in TEMP folder
        elif filetype == "dsm":
            os.makedirs('TEMP/M_', exist_ok=True)
            url = "https://ns_hwh.fundaments.nl/hwh-ahn/ahn4/03a_DSM_0.5m/R_" + str(k.upper()) + ".zip"
            file_path_R =  os.path.join('TEMP/R_',"R_"+str(k.upper())+".tif")
            file_path =  os.path.join('TEMP/R_',"R_"+str(k.upper())+".zip")
            if not os.path.exists(file_path_R):
                print("downloading " + url)
                r = requests.get(url, stream=True)
                with open(file_path, "wb") as f:
                    f.write(r.content) # save file in TEMP folder
                with zipfile.ZipFile(file_path, "r") as zip_ref:
                    zip_ref.extractall("TEMP/R_") # extract contents to TEMP folder
                os.remove(file_path) # remove the zip file after extraction
                print("download complete!")
        elif filetype == "dtm":
            os.makedirs('TEMP/R_', exist_ok=True)
            url = "https://ns_hwh.fundaments.nl/hwh-ahn/ahn4/02a_DTM_0.5m/M_" + str(k.upper()) + ".zip"
            file_path_M =  os.path.join('TEMP/M_',"M_"+str(k.upper())+".tif")
            file_path =  os.path.join('TEMP/M_',"M_"+str(k.upper())+".zip")
            if not os.path.exists(file_path_M):
                print("downloading " + url)
                r = requests.get(url, stream=True)
                with open(file_path, "wb") as f:
                    f.write(r.content) # save file in TEMP folder
                with zipfile.ZipFile(file_path, "r") as zip_ref:
                    zip_ref.extractall("TEMP/M_") # extract contents to TEMP folder
                os.remove(file_path) # remove the zip file after extraction
                print("download complete!")
        os.makedirs('TEMP/RGB_', exist_ok=True)
        url = "https://geotiles.nl/Luchtfoto_2022/RGB_" + str(k.upper()) + ".tiff"
        file_path =  os.path.join('TEMP/RGB_',"RGB_"+str(k.upper())+".tiff")
        if not os.path.exists(file_path):
            print("downloading " + url)
            r = requests.get(url, stream=True)
            with open(file_path, "wb") as f:
                f.write(r.content) # save file in TEMP folder
    print("all maps downloaded")
```

Figure 73 - downloading location data phyton

4.7 Satellite imagery

With the download of the point cloud, it would be nice also to get the colour data to display by using a beamer, for example. The satellite image data can be downloaded like the tiff files described in the chapter above. The only change is the database. Where the point cloud uses the pdock servers, the image is from geotiles (Geotiles, sd). Which are cropped versions of Beeldmateriaal (Beeldmateriaal, sd).

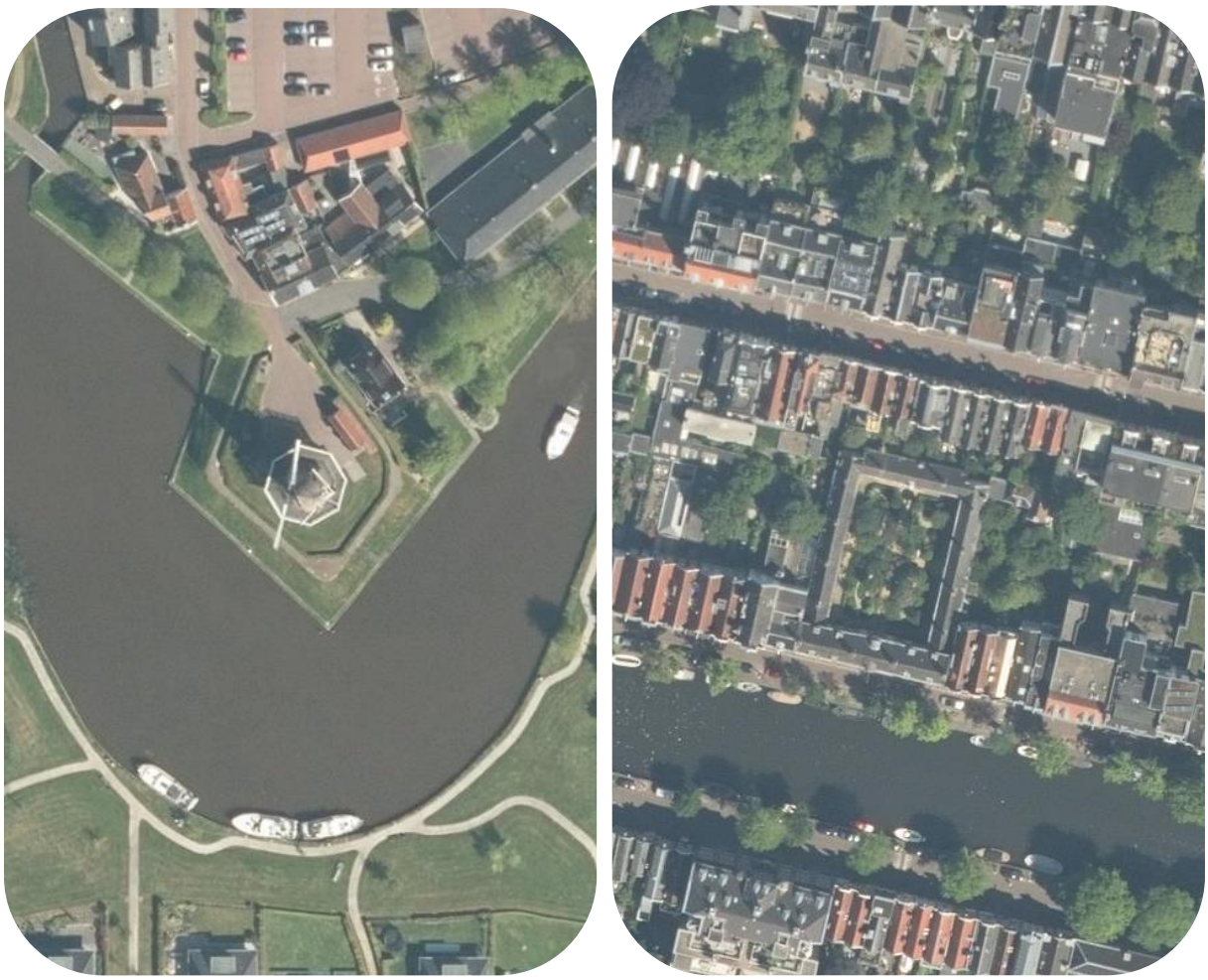


Figure 74 - example cropped satellite imagery, molen de hoop dokkum(left), duitzenhofje Amsterdam (right)

```
633     def export_map(self):
634         print("exporting map")
635         file_path = filedialog.asksaveasfilename(defaultextension='.png', filetypes=[("PNG", '*.png')],
636                                                  title="Choose filename")
637         print(file_path)
638         for kaartblad in self.kaartbladen:
639             print(f"reading {kaartblad}")
640             kaartbladen = pandas.read_excel('kaartbladen.xlsx')
641             row = kaartbladen.loc[kaartbladen['bladnr'] == kaartblad]
642             lo_x, lo_y = row['lo_x'].values[0], row['lo_y'].values[0]
643             tiff_file = "TEMP/RGB_/RGB_"+ kaartblad +'.tiff' # Define the file path to the tiff file
644             # Define the coordinates of the bounding box to crop
645             left = int(max(self.rd4[0] - lo_x, 0)*4)
646             right = int(max(self.rd2[0] - lo_x, 0)*4)
647             bottom = int(max(25000 - (self.rd4[1] - lo_y)*4, 0))
648             top = int(max(25000 - (self.rd2[1] - lo_y)*4, 0))
649 
650             with rasterio.open(tiff_file) as src:
651                 win = Window.from_slices((top, bottom), (left, right)) # Create a window from the pixel coordinates
652                 tiff = src.read([1,2,3], window=win) # Read the RGB data within the window
653                 tiff = numpy.moveaxis(tiff, 0, -1) # Change the order of the dimensions to match the PIL image
654                 tiff = tiff.astype(numpy.uint8) # Convert the data type to uint8
655                 image = Image.fromarray(tiff) # Create an image object from the numpy array
656                 image.save(file_path) # Save the image as a PNG file
657         messagebox.showinfo("Pin Model Generator", f"Image saved on {file_path}")
```

Figure 7533 - save satellite imagery phyton

4.8 Preview

As part of the GUI, a preview is shown to the user to represent the model. This preview also comes with time estimation. A plotting library is needed inside Phyton for a preview to be generated. Luckily Phyton comes with an integrated library called Matplotlib (MPL). This library can also display 3D graphs, of which one is a 3D bar graph, ideal for displaying pins in this case.

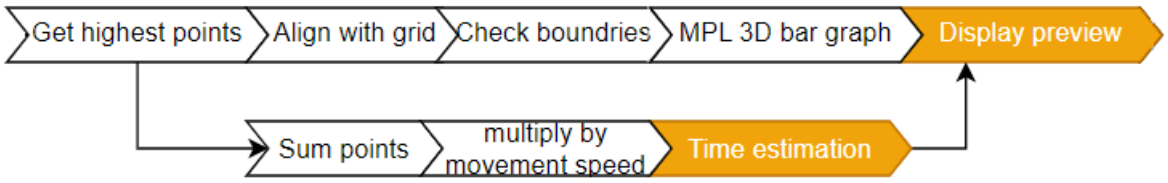


Figure 76 - workflow code preview generation

The time required to generate the model depends on the number of pins and the travel speed of the machine. Below are some calculations to approximate the time needed to create the model.

$$\Delta \max\ time = t_{xy} + t_z = \frac{ab}{zm_{xy}p} + \frac{2c}{m_z z}$$

t = time (s)
m = movement speed (mm/s)
a = shortest length (mm)
b = longest length (mm)
c = sum of z travel (mm)
p = pin size (mm)
z = amount of heads

Example max time 840mm x 1190mm (5mm pins, 150mm/s travel speed all directions):

$$\Delta \max\ time = \frac{840 \times 1190}{150 \times 5} + 2\frac{(\frac{840}{5}) \times (\frac{1190}{5}) \times 100}{150} = 54645\,\text{sec} = 15.2\,h$$

An example is seen in Figure 77.

$$\Delta \max\ time = \frac{840 \times 1190}{150 \times 5} + 2\frac{744610}{150} = 6299\,\text{sec} = 1.7h$$

Time example based on the optimal solution.

$$\Delta \max\ time = \frac{840 \times 1190}{14\ \times 150 \times 3.2} + \frac{2 \times 744610}{300 \times 14} = 148\,\text{sec} + 355\ sec = 8.4\ min$$

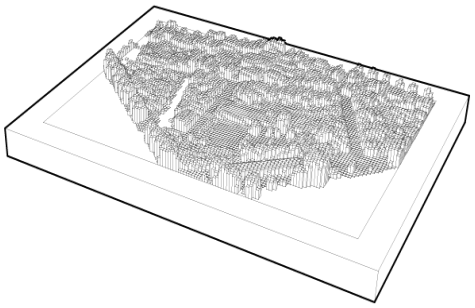


Figure 7734 - example time calculation (5mm pins)

```
535     def time_estimation(self,z_travel):
536         self.values()
537         self.progressbar.set(0.8)
538         estimated_time = str(datetime.timedelta(seconds=int((self.machine_width * self.machine_length)/(int(settings.xy_spee
539         timef = f"{settings.language.estimated_time} {estimated_time}"
540         self.preview_time.configure(text=timef)
```

Figure 78 - time estimation - phyton

```
542     def preview_show(self,dataset):
543         self.tabview.set("Preview")
544         xs,ys,zs = [x[0] for x in dataset],[y[1] for y in dataset],[z[2] for z in dataset]
545         zm = numpy.zeros_like(zs)
546         self.figure_Preview.clear()
547         self.ax_Preview = self.figure_Preview.add_subplot(projection='3d')
548         self.ax_Preview.bar3d(xs, ys, zm, self.pin_size, self.pin_size, zs, shade=True)
549         self.ax_Preview.set_box_aspect((numpy.ptp(xs), numpy.ptp(ys), numpy.ptp(zs)))
550         self.canvas_Preview.draw()
551         self.time_estimation(2*sum(zs))
552         self.progressbar.set(1)
```

Figure 79 - preview generation phyton

The time required for the machine to generate the model thus depends on the number of servos and the movement speed of those. During the research, several recalculations have been made. At first, the machine would seem to take about 8 hours using the stepper motors. After using the servo, this time has been greatly reduced. With the new duet controller, 4 to 6 servos could be connected directly, but this could be expanded to 40, moving up to 40 pins at a time. There is also a big difference in servo moving speed, from 0.2 seconds/60 degrees to 0.06 seconds/60 degrees. This means that 1600 pins/min would easily be achievable. This is not as fast as Shapeshift and Inform but would be more than fast enough to comply with the program of requirements.

4.9 G-code generation

The last part of the code is the g-code generation. With this method, data can be stored in a file to be used multiple times. G-code is the universal coding language for CNC machines. Due to the standardized code, there is an order and text the code should follow. For this, handbooks have been followed. (Smid, CNC Programming Handbook, 2008), (Smid, CNC Control Setup for milling and turning, 2010)

1. create NC file
2. get model information, pin size, machine size, chosen location, etc., and input them as a comment at the top of the file

3. Add home machine line
4. Add for loop
    a. Add xy coordinate on a new line
    b. Next line: Extract z values for given rows of z extensions
    c. Next line: set all servos back to 0
    d. loop
5. Add home command
6. Add end-of-line command

Optimizing the g-code can save much time in the work process of the machine. In Figure 80, the iterations of the program can be seen. In the first example, no optimization is used. The device follows the points inputted by the original code. It goes through the point diagonally and thus uses much time for movement. The points are sorted in the second iteration based on their x and y values. This makes the machine move in only one direction. It reduces the required movement. This code still goes back to the 0 after each row. So the third iteration made the machine go "backward" on all even rows, reducing the unnecessary movement. Note that in all examples, the machine optimizes the movement from the last point in a row to the first in the next row. Due to the working of the g-code, the machine can move in multiple directions simultaneously, reducing the need to follow the grid entirely and reducing the required time even further. The last optimization is done by adding numerous z-axis movements and iterating over every row to see if the next row has a pin to move. This part of the code is run before step 1 in the list above. This is done by reorganizing the list of points given as input. At this point, the code also checks for the last time if there are no points outside the machine's limits.

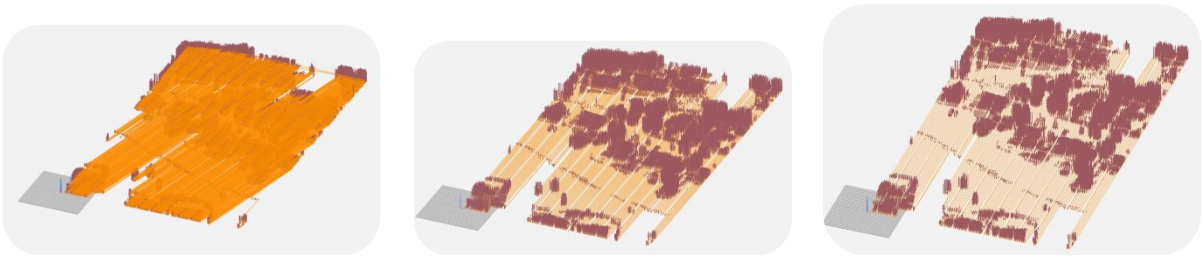


Figure 80 - g-code optimisation

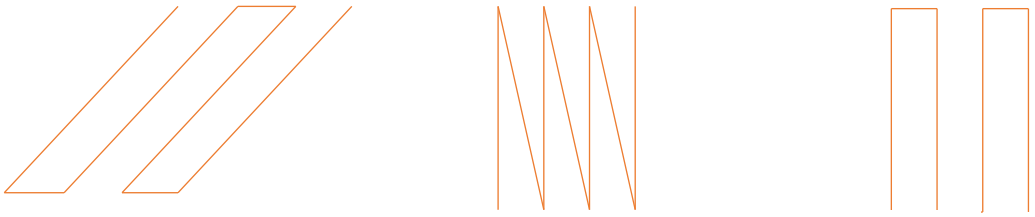


Figure 81 - optimisation visualisation

After testing the initial g-code, there was another problem. Where the machine knows how long it would take for the steppers to move, it did not work for the servos so the servo would stay stuck at 0 degrees. To fix this problem, another line called sleep time was added to the code. With this time, the servo was given a set amount of time to move. The time given is based on the servo's movement speed, which is 0.15sec/60 degrees, so .45 seconds for a full rotation. As the servo does not always go as fast as planned, another 10% is added to ensure it can do the whole movement.

5. Combined evaluation

Cases have been generated and tested to test if the machine functions as described. In the first test, the g-code was tested on a known working CNC to test if it would follow the path given in the g-code.

Another test that has been done is the wear test on the EPDM inside the matrix. In this test, the matrix was placed under a CNC machine with a code to go 1000 times up and down with a metal rod through the EPDM layer. After the run, another rod was pushed through the EPDM to see the friction difference. Although the EPDM was a bit looser on the side where the wear test happened, it did not loos to much friction to the point where it could no longer hold a pin.

The 3D test was to check the proper working of the software. As shown in Figure 82(left), a location was selected, and the points were generated. The preview showed the correct location and an estimated time based on the given speeds of the Known CNC machine. After the generation of the g-code, it first had to check in online software to check to code. After that, the code was timed on the known CNC machine.

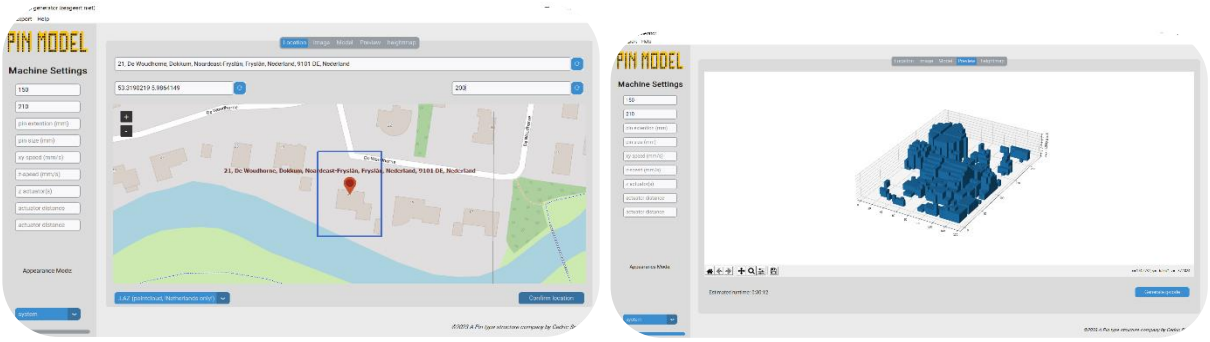


Figure 82 - location selection and preview

Selected location
Machine settings:
150x210 (test frame size)
XYZ movement: 40mm/s test rig Ooznest workbee CNC

Predicted worktime: ~30min 12 sec

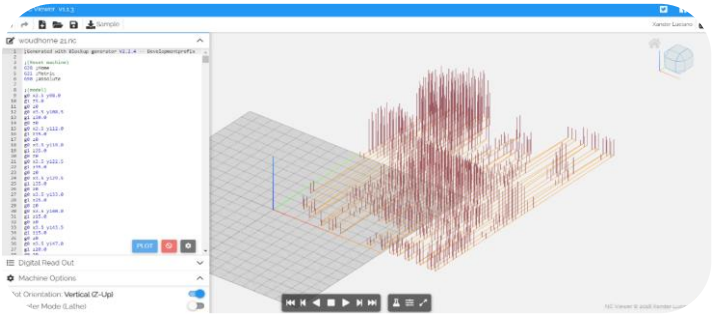


Figure 8335 - g-code viewer

Actual work time: 25min 24 sec

Actual work speed: 2504mm/min = 41,73mm/sec, so faster movement

Another possibility to be faster: the production takes the entire XY movement as a small part of the time. This time is reduced significantly on a small scale and with not all pins to be moved.

The next step was the generation of all the case studies in the code. This is to check if the code works as expected.

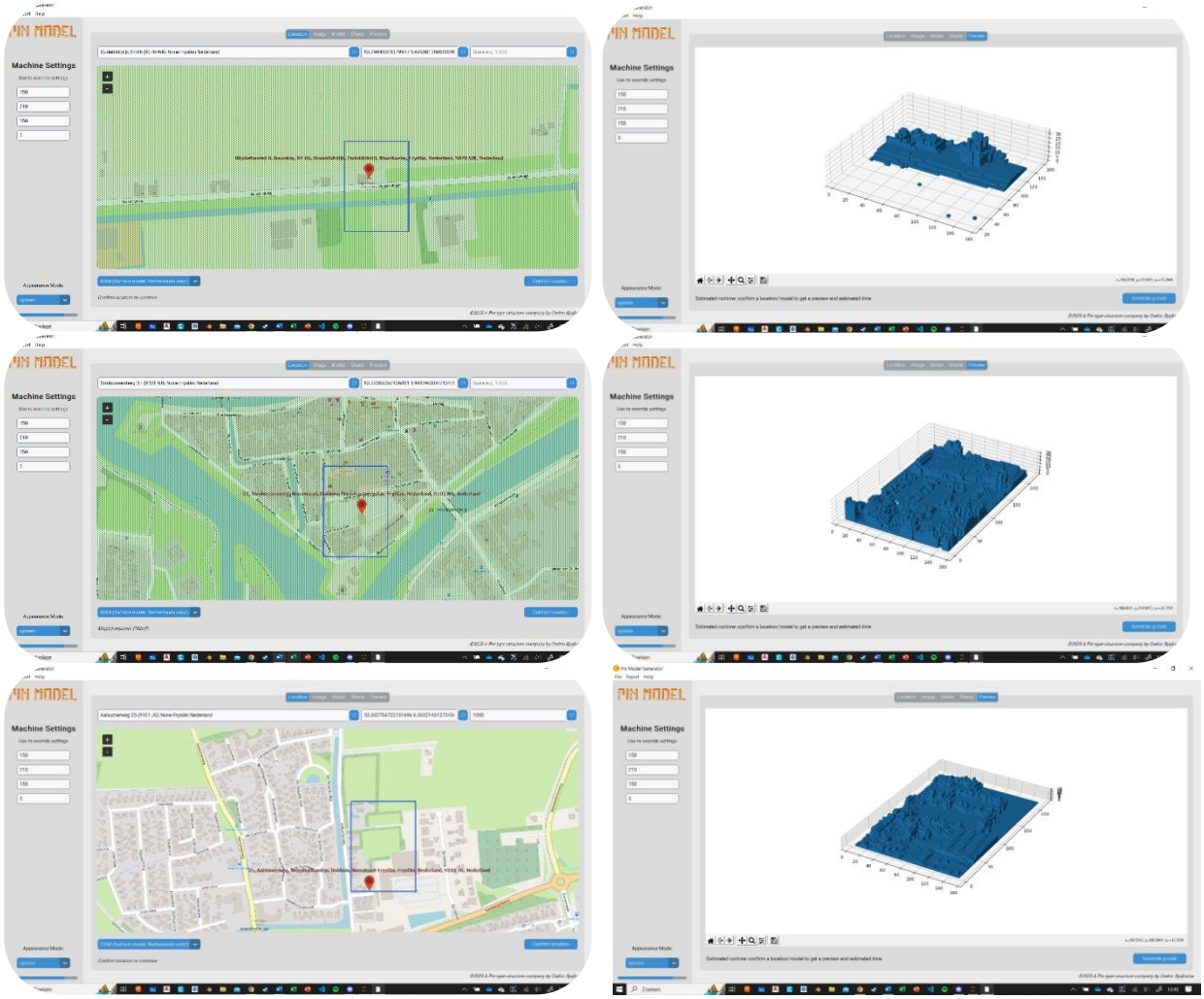


Figure 84 - case studies in code

In addition to the specified case studies, various exceptional scenarios were also examined. These included locations between two map tiles on small and large scale and tall buildings. The method of calculation implemented in the code effectively handled the locations between two tiles, posing no issues. Also, the tall building was generated with no problem, displaying the church of Delft (the second tallest church tower in the Netherlands). However, when attempting with certain scales, problems arose. The maps utilized for calculation did not always provide enough data points for recalculation, resulting in unmoving strips on the map, as shown in Figure 85.

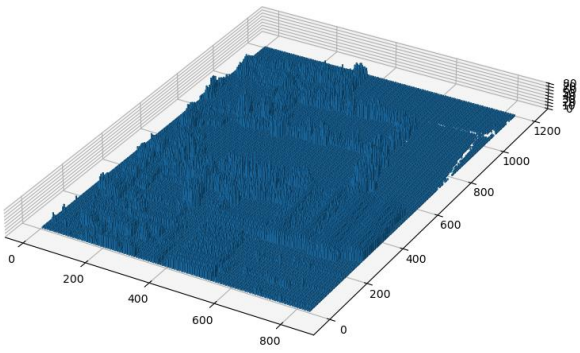


Figure 85 - artifacts due to scale

Once the code determined the optimal path, it was executed on the prototype machine. The prototype allowed for simulating models of dimensions 150mm x 210mm. The models were generated in less than 10 minutes by utilizing two functioning servos, a very promising result.

6. Conclusion and Reflection

6.1 Conclusion

In conclusion, this research highlights the potential of pin models as a cost-effective and versatile physical modeling technique in architectural and urban/rural design. It addresses the limitations of current models, which tend to be expensive and have low resolution, by proposing a solution that offers higher resolution and reduced costs. The significance of physical models in architecture and teaching is reaffirmed, as they provide a tangible representation of design concepts and facilitate effective communication.

The device's usability in this research has focussed on architects and urban planners engaged in mass studies, addressing their unique needs in generating accurate and efficient models. The suggested device size and proportions, such as an A0 sheet or a 16:9 ratio, offer a practical working area for mass modeling but could be adjusted to the need of customers.

The device comprises three key components: visual pins, a state layer of EPDM/rubber that holds the pins in place, and a moving mechanism. The state layer's holding power, facilitated by friction, is sufficient for the intended application and differs from other models. Optimizing the electronics and servo configuration impacts the device's effectiveness, directly influencing its operating speed. With 40 pins per run at 1600 pins per minute, it is slower than the direct drive from the inform and shapeshift, but fast enough to meet the program of requirements.

While the visual pins are recognized as the most expensive component of the machine, the overall affordability of the device relies on the availability and pricing of these pins. The developed code, designed with user-friendliness in mind, streamlines the workflow by automatically gathering data from sources like AHN (Actueel Hoogtebestand Nederland) via the internet. This efficient process enables the seamless generation of pin models. Additionally, the device exhibits versatility by accepting inputs such as location data, 3D models, point clouds, and images, accommodating different user requirements. Furthermore, including map display functionality enhances the visual representation by providing valuable contextual information. The research findings suggest that the code offers a competitive advantage over traditional model-making techniques regarding speed and reusability. Consequently, it presents an attractive option for companies heavily involved in physical model production or those seeking to expand their design capabilities and gather information effectively.

Further research is required to advance this field of study. Areas that require attention include investigating the availability and cost of visual pins to determine the economic viability of the device. Additionally, exploring alternative materials and design modifications could optimize the device's performance and capabilities. Future research should also focus on expanding the device's applicability to other trades beyond architects and urban planners. Furthermore, exploring different device sizes and proportions can provide options for various project requirements and contexts. And lastly, more research could be used to convert digital data to g-code.

In conclusion, the findings of this research demonstrate the potential of pin models as a valuable physical modeling technique. By addressing the limitations of current models, optimizing design, and streamlining the workflow through code, architects and urban planners can benefit from cost-effective, high-resolution models that enhance their design process and decision-making.

6.2 Reflection

Throughout my graduation project, I had a challenging yet rewarding journey. The project pushed the boundaries of physical and digital possibilities, necessitating a multidisciplinary approach encompassing mechanical and electrical engineering, coding, and design. While the project sometimes felt disconnected from Building Technology, there were significant similarities, such as model making, design approaches, and coding integration as part of digital manufacturing. Without the knowledge acquired during the study and the main background of Building technology, this topic would not have been researched.

As anticipated, a substantial amount of time was dedicated to creating the prototype, and the process was not without its obstacles. However, I particularly enjoyed the practical, hands-on exploration of this research. It allowed me to engage in various aspects simultaneously, dividing my work into research, writing, and physical prototyping. This approach, starting with research and developing a program of requirements, greatly aided the subsequent research-by-design phase.

The chosen methodology and time frame, as outlined in the introduction, proved to be fitting for the overall process. The early research phase laid a solid foundation, significantly influencing the subsequent research by-design phase. While the coding aspect progressed faster than expected, thanks to the utilization of AI generation and the availability of point clouds and datasets. The prototyping phase unfolded more gradually, aligning with my initial expectations. I maintained informal contact with architectural firms throughout the research, benefiting from feedback.

During my project, I faced several notable challenges, including slow deliveries and the expensive nature of materials. Despite these obstacles, I remained determined and managed to source the necessary components, ultimately creating a functional prototype. The process of learning to code for this research presented a significant learning curve for me, considering my limited prior experience in this field. Data conversion emerged as a particularly challenging aspect of the coding process. It was converting data from one file type to another that introduced complexities that increased the margin of error. Furthermore, the large volume of data that needed to be stored and converted added to the computational difficulties, making it demanding for the computers to handle the task effectively.

In electronics, I struggled with compatibility among reused parts. Motor malfunctions, damaged power converters, and missing shields complicated the process of getting the electronics to function correctly. The pin assembly was challenging, as the initial design required excessive pins, necessitating a reduction. Finding suitable pin materials proved time-consuming, as wood & PLA proved unsuitable.

Despite these hurdles, I take pride in my final product. The mechanism of the machine now operates smoothly, and the precise tolerances of the HDPE material ensure that the device fulfills its intended function, at a relative low cost. This project has equipped me with invaluable problem-solving skills, fostered creativity, and instilled perseverance. I eagerly anticipate applying these lessons learned to future endeavors.

However, this is not the end of the journey. The working prototype, with its efficient code, demonstrates the promise of this machine. Questions now arise regarding scalability, affordability, and alternative applications. While the current research concludes here, the future holds endless possibilities for further exploration and development.

In conclusion, the graduation project has been a fulfilling experience. The multidisciplinary nature of the work, challenges, and lessons learned have shaped me personally and professionally. I appreciate the growth opportunities and am excited to see how the knowledge and skills gained during this project will contribute to my future pursuits.

References

A, M. (2007). Computational Morphogenesis Integral Form Generation and Materialization Processes. *3rd Int'l ASCAAD Conference on Em'body'ing Virtual Architecture*.

Agnieszka, F., Taraszkiewicz, S., & Taraszkiewicz, A. (2020). The role of traditional architectural models in the first stages of education. *World Transactions on Engineering and Technology Education*, pp. 177-182.

Beeldmateriaal. (n.d.). *Beeldmateriaal*. Retrieved from https://www.linkedin.com/posts/beeldmateriaal_opendata-beeldmateriaal-bm-activity-6747095018730725376-UMU_/

Bergstrom, M. (2022). *cities in miniature*. Retrieved from wurlington-bros: https://www.wurlington-bros.com/DC/minicities.html

Berkel, B., & Bos, C. (2006). *Design Models - Architecture, Urbanism, Infrastructure.* London: Thames and Hudson.

Berteau, J. (1994). *U.S. Patent No. 5,330,343.*

Boers, S. H. (2006). *Optimum forming strategies with a 3D reconfigurable die.* Eindhoven.

Cannaerts, C. (2019). *Models of / Models for Architecture - Physical and Digital Modelling in Early Design Stages.* Schaarbeek.

Collette, W. N., Piccioli, D. P., & Krishnakumar, S. M. (1993). *U.S. Patent No. 5,255,889.*

Downton, P., Ostwald, M., Mina, A., & Fairley, A. (2007). *Homo Faber - Modelling Architecture.* Sydney: Archadia Press.

Duet 3D. (n.d.). Retrieved from https://docs.duet3d.com/

Follmer, S., Leithinger, D., Olwal, A., Hogge, A., & Ishii, H. (2013). *inFORM: Dynamic Physical Affordances and Constraints.* Cambridge: MIT media.

Galizia, F. G., Elmaraghy, W. H., Elmaraghy, H. A., Bortolini, M., & Mora, C. (2019). *The evolution of molds in manufacturing: from rigid to flexible.* Researchgate.

Geotiles. (n.d.). Retrieved from https://geotiles.nl/

Gibson, I., Kvan, T., & Ming, L. W. (2002). Rapid prototyping for architectural models. *Emerald*.

Globa, A., Donn, M., & Twose, S. (2012). Digital To Physical: Comparative Evaluation Of Three Main CNC Fabrication Technologies Adopted For Physical Modelling In Architecture. *international journal of architectural computing*, pp. 461-480.

Google. (2022). *Maps*. Retrieved from Google: https://www.google.com/maps

Haas, E. G., & Kesselman, M. (1996). *U.S. Patent No. 5,546,784.*

Haas, E. G., Schwarz, R. C., & Papazian, J. M. (2000). *U.S. Patent No. 6,089,061.*

Haas, E. G., Schwarz, R. C., & Papazian, J. M. (2003). *U.S. Patent No. 6,578,399.*

Hoffman, P. L. (1998). *U.S. Patent No. 5,846,464.*

Kadaster. (n.d.). *Basisregistratie Adressen en Gebouwen (BAG) (Dutch)*. Retrieved from Kadaster: https://bagviewer.kadaster.nl/lvbag/bag-viewer/#?geometry.x=160000&geometry.y=455000&zoomlevel=0

Kitchenplannertable. (n.d.). *touchscreen-ontwerptafel.* Retrieved from kitchenplannertable: https://www.kitchenplannertable.com/nl/touchscreen-ontwerptafel/

Kvan, T., & Ming, L. W. (2002). Rapid prototyping for. *Emerald*, pp. 91-99.

Laskowski, J. S. (1998). *U.S. Patent No. 5,796,620.*

Morita, M. (1993). *U.S. Patent No. 5,187,969.*

Munro, C., & Walczyk, D. (2007, June). Reconfigurable Pin-Type Tooling:. *Journal of Manufacturing Science and Engineering*, p. 551.

Openbuilds Design. (2023). *Openbuilds*. Retrieved from Openbuilds: https://openbuilds.com/

OpenStreetMap. (2022). *OpenStreetMap*. Retrieved from OpenStreetMap: https://www.openstreetmap.org/

ORF, D. (2013). *How It Works: MIT's inFORM Dynamic Shape Display*. Retrieved from popularmechanics: https://www.popularmechanics.com/technology/design/a9847/how-mits-inform-dynamic-display-works-16222829/

Panero, J., & Zelnik, M. (1979). *Human Dimension and Interior Space.*

Papazian, J. M., Haas, E. G., Schwarz, R. C., Nardiello, J. A., & Melnichuk, J. (2001). *U.S. Patent No. 6,209,380.*

Papazian, J. M., Nardiello, J. A., Schwarz, R. C., & Melnichuk, J. (2002). *U.S. Patent No. 6,363,767.*

Pommer, R. (1981). *Ideas as Model.* New York: Rizzoli.

Redwood, B., Schöffer, F., & Garret, B. (2017). *The 3D Printing Handbook: Technologies, design and applications.* 3DHubs.

Schroeder, T., & Stevenson, R. (1998). *U.S. Patent No. 5,738,345.*

Siu, A. F., Gonzalez, E. J., Yuan, S., Ginsberg, J. b., & Follmer, S. (2018). *shapeShift: 2D Spatial Manipulation and Self-Actuation of.* Montréal: Stanford University.

Smid, P. (2008). *CNC Programming Handbook.* New York: Industrial Press.

Smid, P. (2010). *CNC Control Setup for milling and turning.* New York: Industrial press.

Smith, A. C. (2004). *Architectural model as machine: a new view of models from antiquity to the present day.* Oxford: Architectural Press.

Stellingwerff, M., & Koorstra, P. (2013). Model & scale as conceptual devices in architectural. *EAEA-11 conference*, pp. 491-498.

Sullivan, E. V., Haas, E. G., Schwarz, R. C., Kesselman,, M., Peck, A. N., & Papazian, J. M. (2000). *U.S. Patent No. 6,012,314.*

Todoroki , M., Imazu , H., Nomura, H., Yamaguchi, N., Zama, J. A., Ishibashi, K., & Yamamoto, K. (1993). *U.S. Patent No. 5,253,176.*

TU Delft. (2015, June 17). *Bouwtechnisch Tekenen: Tekentechnieken (dutch)*. Retrieved from BK WIKI: http://wiki.bk.tudelft.nl/bk-wiki/Bouwtechnisch_Tekenen:_Tekentechnieken#Schaal

Ultra Precision Motion. (2015). *Automan.* Retrieved from Automated Manufacturing Process Integrated with: https://www.nottingham.ac.uk/ifam/documents/dtmi2015/3-6-automan.pdf

Umetsu, S., & Toshihiko, M. (1993). *U.S. Patent No. 5,192,560.*

Ventio. (n.d.). *Machine Builder*. Retrieved from https://vention.io/

Vink, M. G., & Koorstra, P. A. (2020). Beeldend onderzoek: Tekening en model (dutch). *Inzicht*, pp. 155-168.

Walczyk, D. F., & Hardt, D. E. (1998). Design and analysis of reconfigurable discrete dies for sheet metal forming. *Journal of Manufacturing Systems*.

Wang, Z. (2010). *Rapid manufacturing of vacuum forming components utilizing reconfigurable screw pin tooling.* University of Nottingham,.

Weinstock, M., Burry, M., Hensel, M., & Menges, A. (2006). *Morpho-Ecologies:Towards a Discourse of Heterogeneous Space in Architecture.* London: AA Publications.

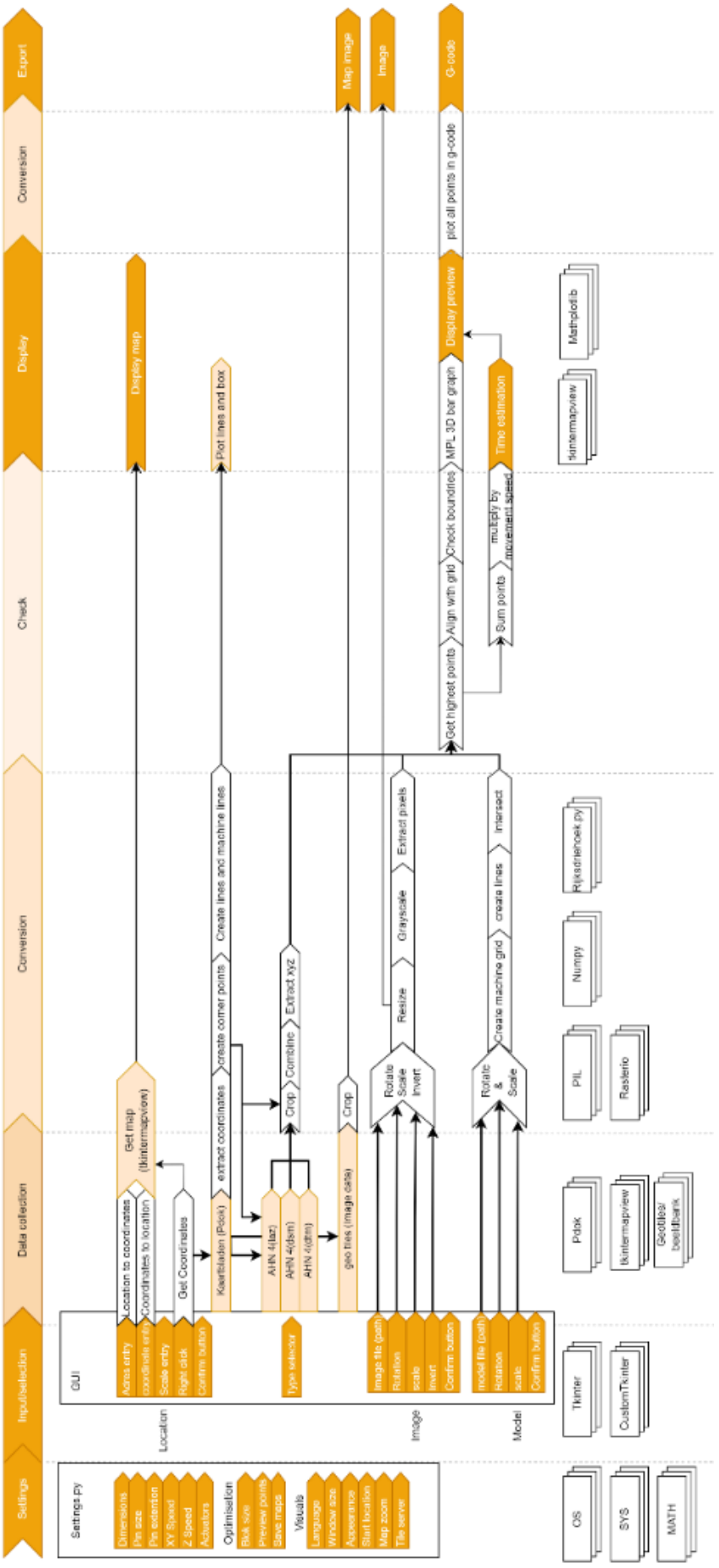
Whitacre, F. E. (1971). *U.S. Patent No. 3,559,450.*

WILSON, M. (2018). *A Computer Mouse For The Year 3000*. Retrieved from Fastcompany: https://www.fastcompany.com/90169965/shapeshift-a-computer-mouse-for-the-year-3000

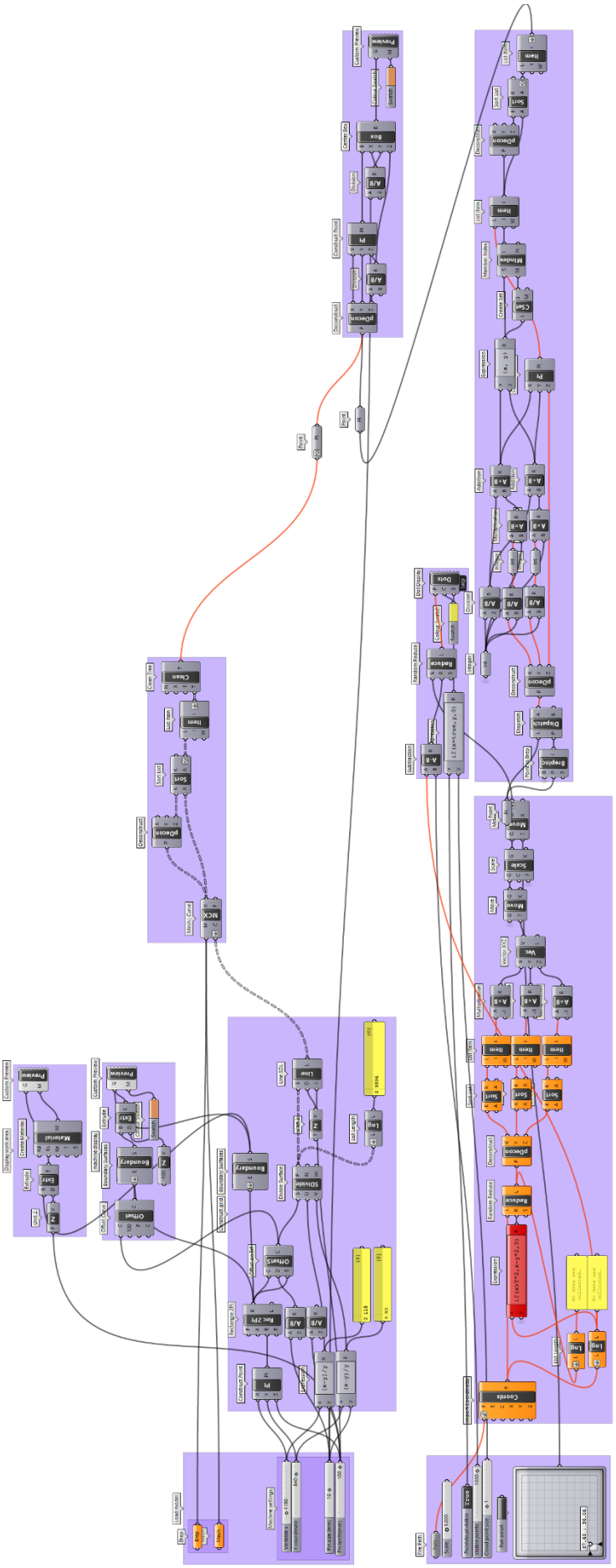
Zhang, K., & Follmer, S. (2018). *Electrostatic Adhesive Brakes for High Spatial Resolution Refreshable.* San Francisco, USA: Haptics Symposium.

Appendix

Code overview

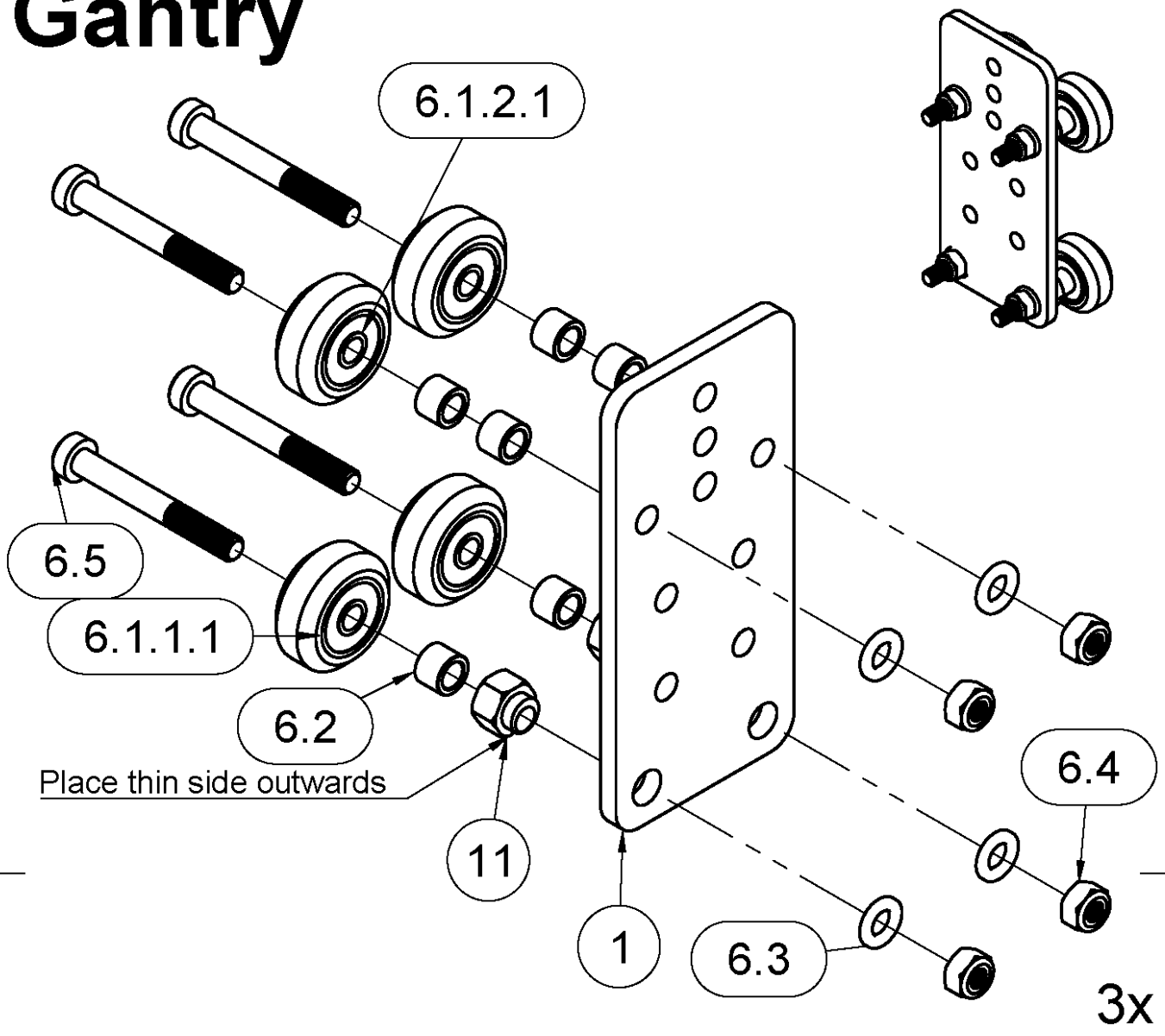


Code overview - Grasshopper



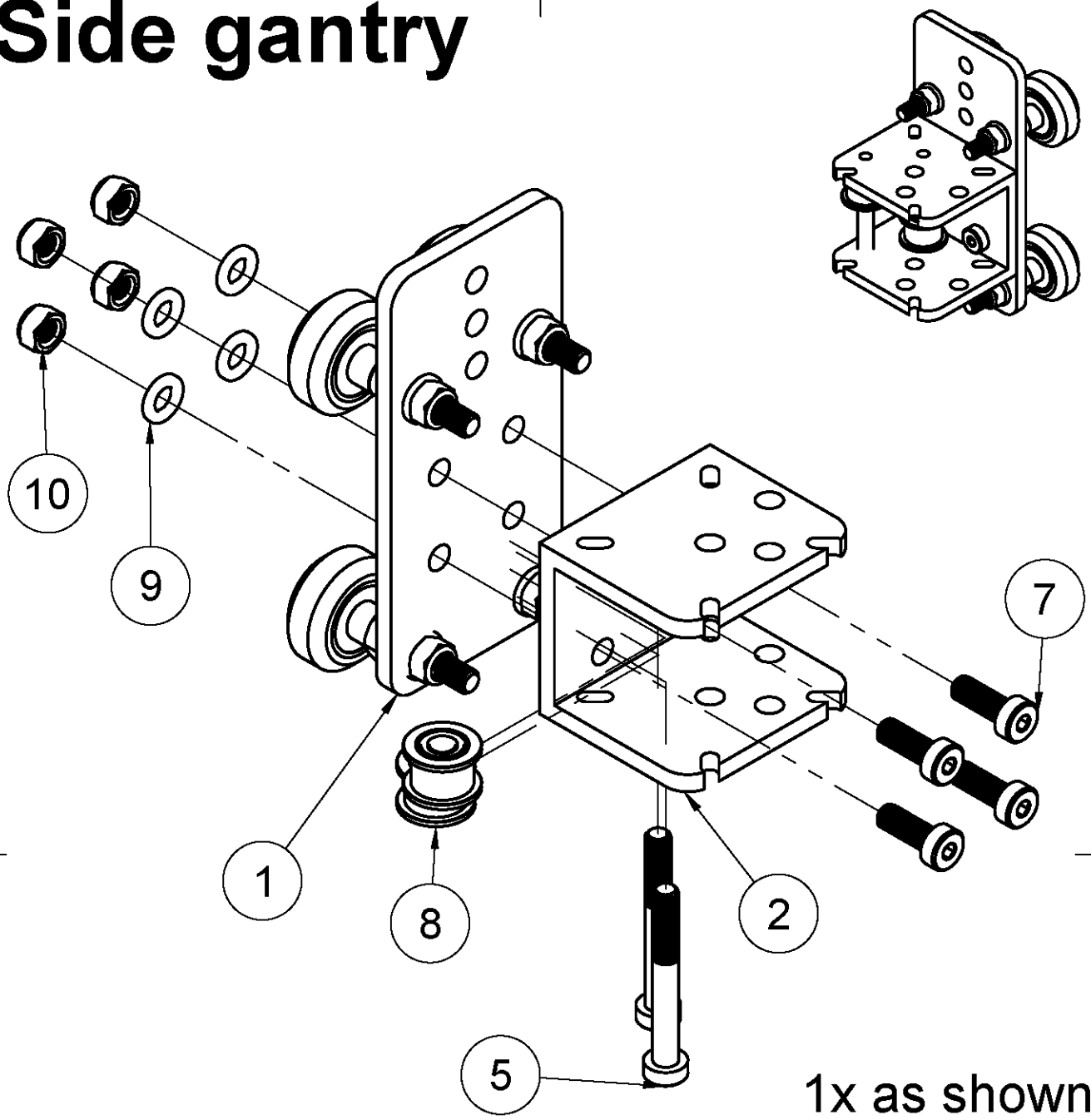
Assembly manual Mechanism

Gantry



Parts List		
Item	Qty	Part Number
1	1	Gantry plate
6	4	Wheel assembly
6.1	1	V-Wheel Kit
6.1.1	1	Solid V-Wheel
6.1.1.1	1	Component5
6.1.1.1.1	1	External1
6.1.2	2	Ball Bearing 625 2RS 5x16x5
6.1.2.1	1	Component4
6.2	2	94669A042_Aluminum Unthreaded Spacer
6.3	1	Wheel assembly v5 (1)
6.4	1	Wheel assembly v5 (2)
6.5	1	93070A133_Alloy Steel Low-Profile Socket Head Screw
11	2	Eccentric Spacer 6mm

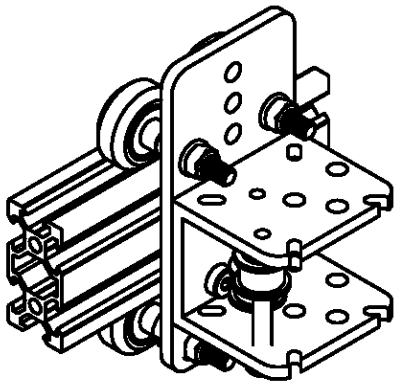
Side gantry

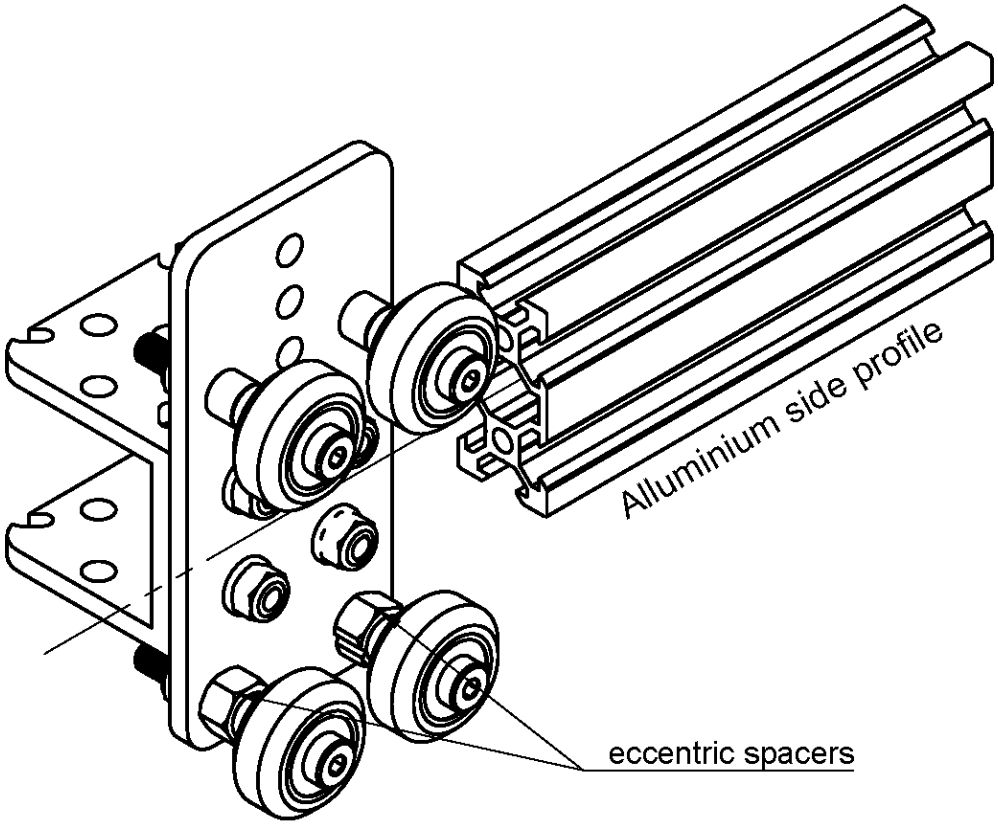


1x as shown
1x mirrored

Parts List		
Item	Qty	Part Number
1	1	Gantry plate
2	1	Corner and motor mount
5	2	93070A133_Alloy Steel Low-Profile Socket Head Screw
6	4	Wheel assembly
7	4	93070A124_Alloy Steel Low-Profile Socket Head Screw
8	2	3693N14_High-Strength HTD Timing Belt Idler Pulley
9	4	Wheel assembly v5 (1)
10	4	Wheel assembly v5 (2)
11	2	Eccentric Spacer 6mm

Side Gantry





2x

First make sure that the wheels can still rotate freely.

Use a spanner to adjust each eccentric spacer so that this the short wall is facing down. Doing this maximizes the gap between the top and bottom row of wheels.

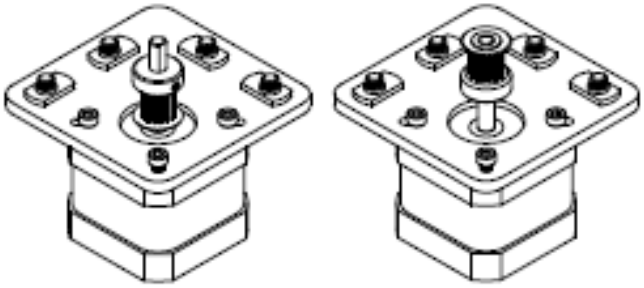
Run the piece of aluminium extrusion inbetween the two rows of wheels. Initially, there may be a small amount of play between the extrusion and wheels.

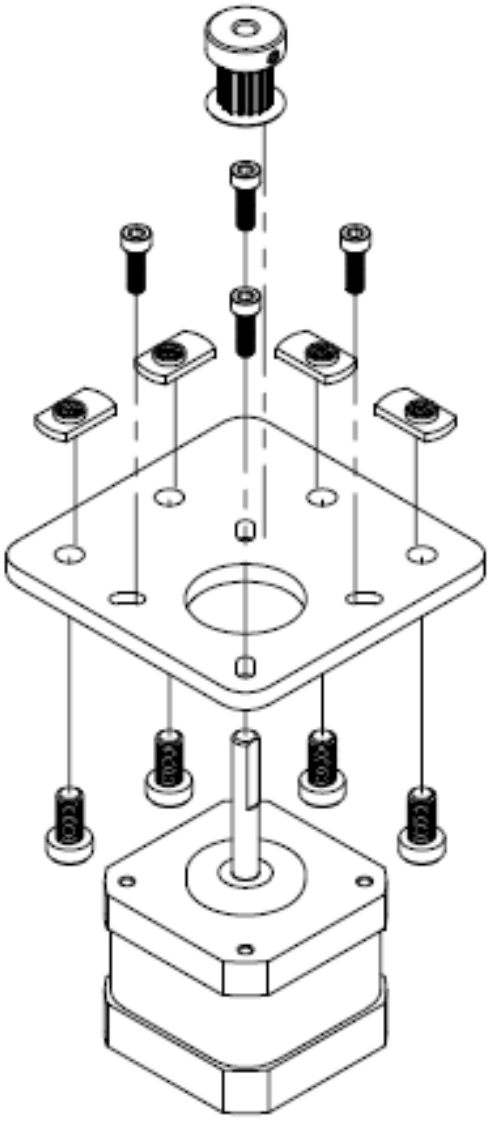
Starting with one pair of wheels, adjust both eccentric spacer down onto the extrusion until there is a small amount of friction between both wheels and the extrusion.

When adjusting the pair of eccentric-spacer ideally they should be adjusted identically. However, sometimes one will need to be adjusted slightly more than the other to get both wheels engaged with the extrusion.

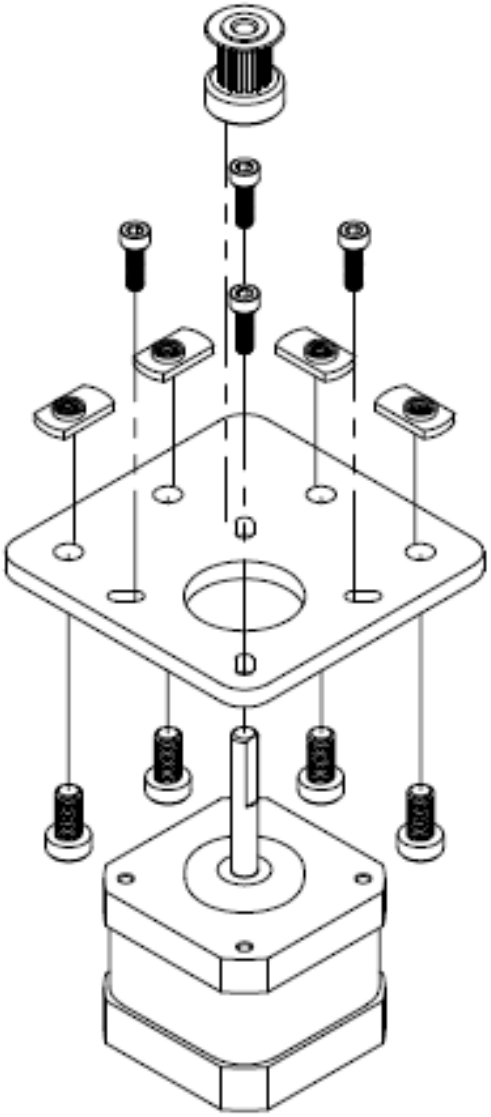
Slide the extrusion back and forth through the wheels. This should require a small amount of force, and all wheels should spin as it rolls. Also check there is no wobbling of the extrusion. Once happy, double check the tightness of the M5-Nyloc Nuts and the rotation of the wheels.

Motor Mount





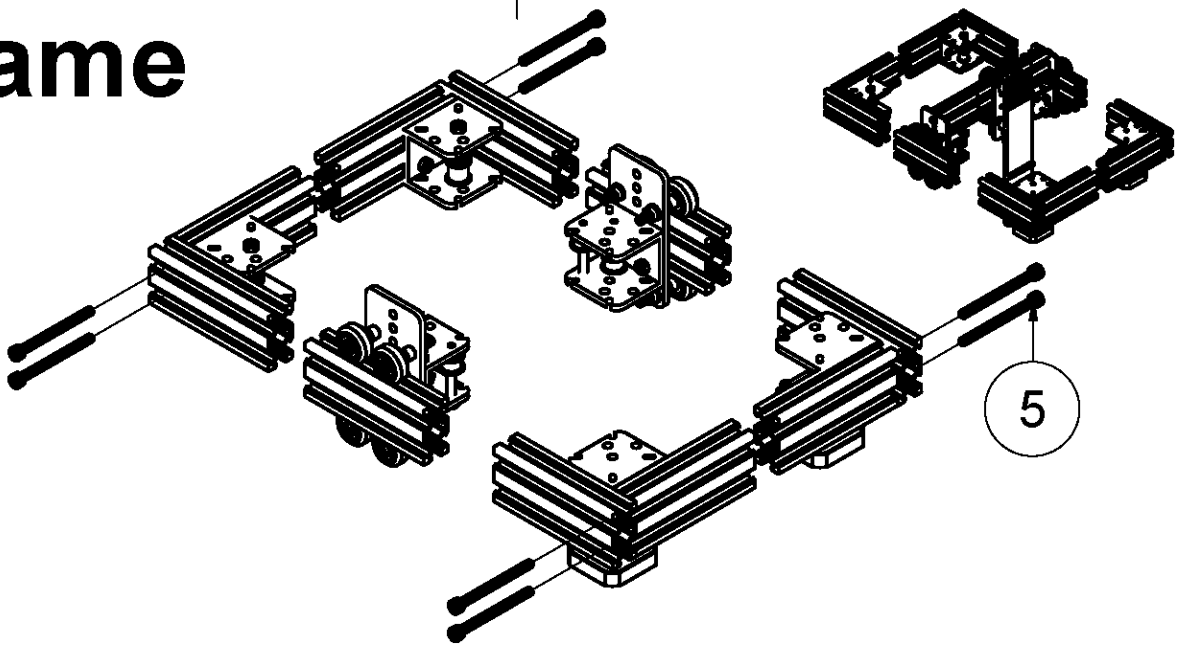
1X Motor mount

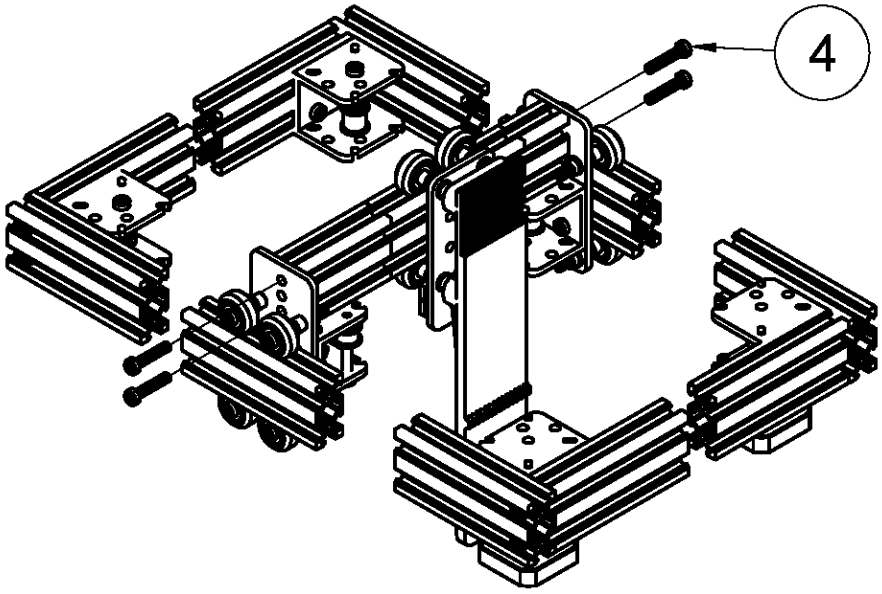


1X Motor mount (inverted pulley)

Parts List		
Item	Qty	Part Name
1	2	Motor mount plate
2	2	Stepper motor
3	2	Toothed Pulley
4	8	M3 - 10mm bolt
5	8	M5 - 20mm bolt
6	8	T-slot nut

Frame





make sure alle the bolt are propperly tightend of all the sub assembly's changis this later can be a lot harder
when bolting the frames together make sure all the parts are facing the right direction.

The motors and pullies are facing inwards, The gantries are plased on the extrucions, The z gantry is facing the motors, The pullies on the side gantry are on the pulley side

Tighten the bolts into the frame, but not to tight as it is tapped in aluminium

Parst List		
Item	Qty	Part Number
4	4	m5 - 60mm low profile hex bolt
5	8	m5 - 60mm hex bolt
	2	side assamblies
	1	motor mount assambly
	1	pulley mount assambly
	1	Z gantry assembly

Assembly manual Electronics

⚠⚠⚠Warnings⚠⚠⚠

Never connect or disconnect anything to the electronics board(s) when power is applied!
Exception: you may connect or disconnect the USB cable with power applied, but see the important notes on using a USB connection. Also, see USB ground loops.

Take great care to connect the + and - leads from the power supply the right way to the VIN terminal block!
Use the supplied power supply or similar, and ensure enough wattage for your system. Do not exceed 32V VIN input voltage!

When the stepper motors are connected to the electronics board, do not move any parts by hand that make the motors rotate rapidly!
For example, if you need to move the gantry by hand, do it **SLOWLY**. Rapid rotation of the stepper drivers will generate enough voltage to power them up uncontrolled and could generate enough to exceed their rated voltage.

Do not insert an SD extender cable into the built-in SD card socket!
These cables frequently damage the contacts inside the SD card socket. Even when they don't, they do not work well at the high SD card transfer speeds used by the Duet.

When mounting the electronics board(s), if you use metal screws, make sure they don't touch any electronic components or solder pads!
Use the supplied hardware and follow the instructions.

Be very careful if you use a multimeter to measure voltages on the board, especially on the fan connectors!
The probes may slip and short against each other or into other components. If you short the two pins of a fan connector together, you can damage the boards.

Do not rely solely on the Duet electronics and firmware to guard against excessive temperatures in your system
Despite the advanced protection features in Duet firmware, electronics and firmware can fail unexpectedly. We do not recommend running the system unsupervised!

If you use a USB connection to the Duet, mitigate the effect of ground loops!
See USB ground loops.

For more info and the latest warnings, see:
https://docs.duet3d.com/User_manual/Overview/Getting_started_Duet_3_MB6HC

⚠USB ground loops⚠
Connecting your mainboard to a PC using a USB cable usually creates a ground loop. This ground loop can be problematic unless precautions are taken.

Why is there a ground loop?
When a PC is plugged into the mains, there is a connection between the ground pin of the mains plug and the ground pin on the USB connector. This is usually the case when you use a laptop with the mains charger connected. When you power your electronics from a mains-powered 12V or 24V power supply, again, there is usually a connection between the ground pin of the mains plug and the negative output of the PSU(power supply unit). This is always the case for ATX power supplies and may be the case for other supplies.

The negative VIN input to the Duet is connected to the ground pin on the USB connector. When you connect a USB cable, the cable connects the ground pin of the USB connector on the PCB to the ground pin of the USB connector on the Duet, which completes the loop.

Problems that a ground loop can cause
The USB cable and mains ground connections form an alternative path for current to flow between the Duet and the negative power supply output. If the connection between the negative terminal of Duet's VIN terminal block and the negative output from the power supply is weak, then this alternative path will be used. The USB cable will become hot, and you risk damaging the USB ports on the PC and the Duet.

If the ground connections of the mains supplies of the Duet and the PC are at different potentials, for example, because of a mains fault or because other equipment connected to the mains supply has significant ground current, a large current will flow through the USB cable ground wire.

Other devices powered from the mains may create ground transients. These will flow through the USB cable and may cause the Duet to reset or perform abnormally in other ways.

Mitigating the problems
Don't connect the system to a PC via USB when you don't need to. The Duets all have Ethernet or WiFi connections, so USB is generally used only when debugging.

If you do need to use a USB connection, power the PC and the Duet's power supply and nothing else from either a double mains socket or from a separate mains distribution block.

Ensure that the cable between the power supply and the negative VIN terminal on the Duet is rated adequately for the current and secure at both ends. Check that the VIN terminal block screws on the Duet remain tight.

Pass the USB cable a few times through a ferrite ring, or use a USB cable that has a ferrite ring fitted already, or buy a split ferrite bead and clamp it over the lead. This mitigates the effect of ground transients.

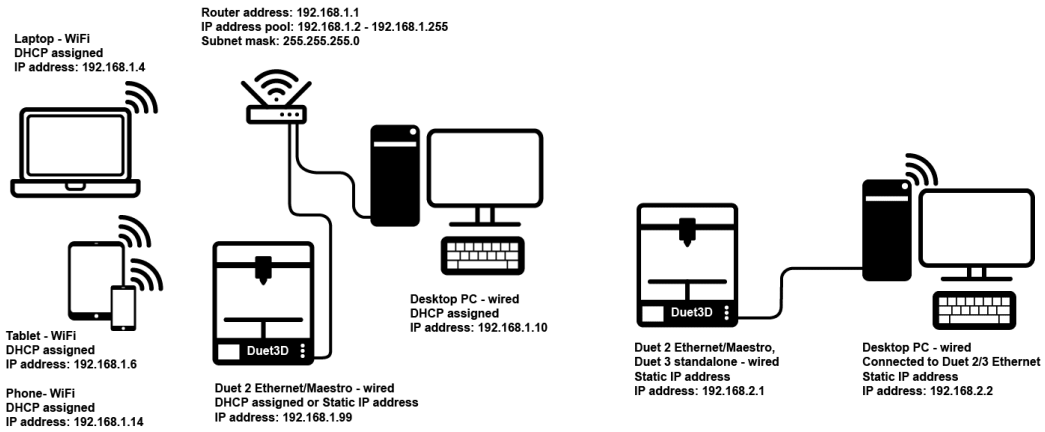
Avoiding USB ground loops
- Don't use a USB connection. You don't need one in regular use because you can perform all operations (including uploading GCode files and changing settings) via the web interface.
- Connect the electronics via USB to a laptop running off batteries, with the charger not connected.
- Use a USB isolator module.
- Disconnect the power supply negative output from the mains ground. (not recommended)

Manual Software

Getting connected

This guide covers getting a basic USB connection to your Duet, setting it up to be connected to your network, and connecting to the Duet Web Console using a browser.

For the most up-to-date instructions, go to:
https://docs.duet3d.com/User_manual/Machine_configuration/Networking
Follow the instructions for the standalone Duet 3 Mainboard 6HC(Ethernet)



After you have done the Getting connected, go to the ip address defined in the previous step.
It would be wise to save the IP address in your browser for quick access every time.

When the machine is correctly connected to the network, and you can log in using a browser, we can now upload the files.

Go to the settings tab to the left, press upload files, and upload the files(settings files) in the settings folder.

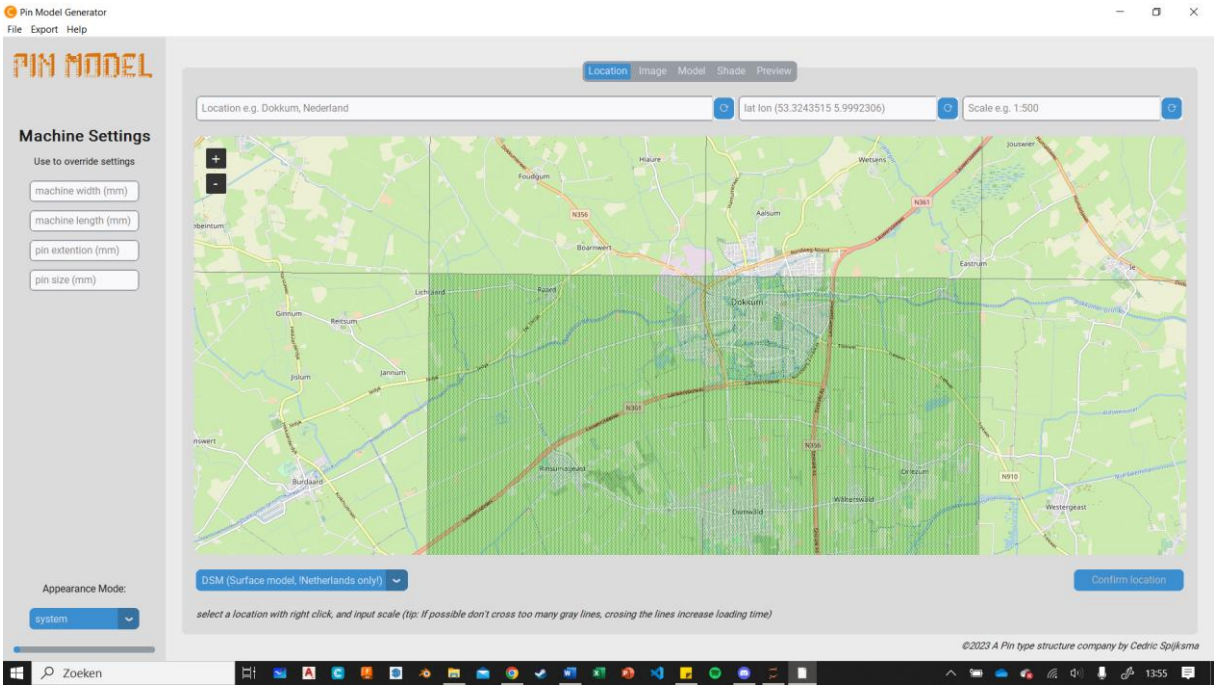
Manual Creating g-code

Step 1: (first time only)

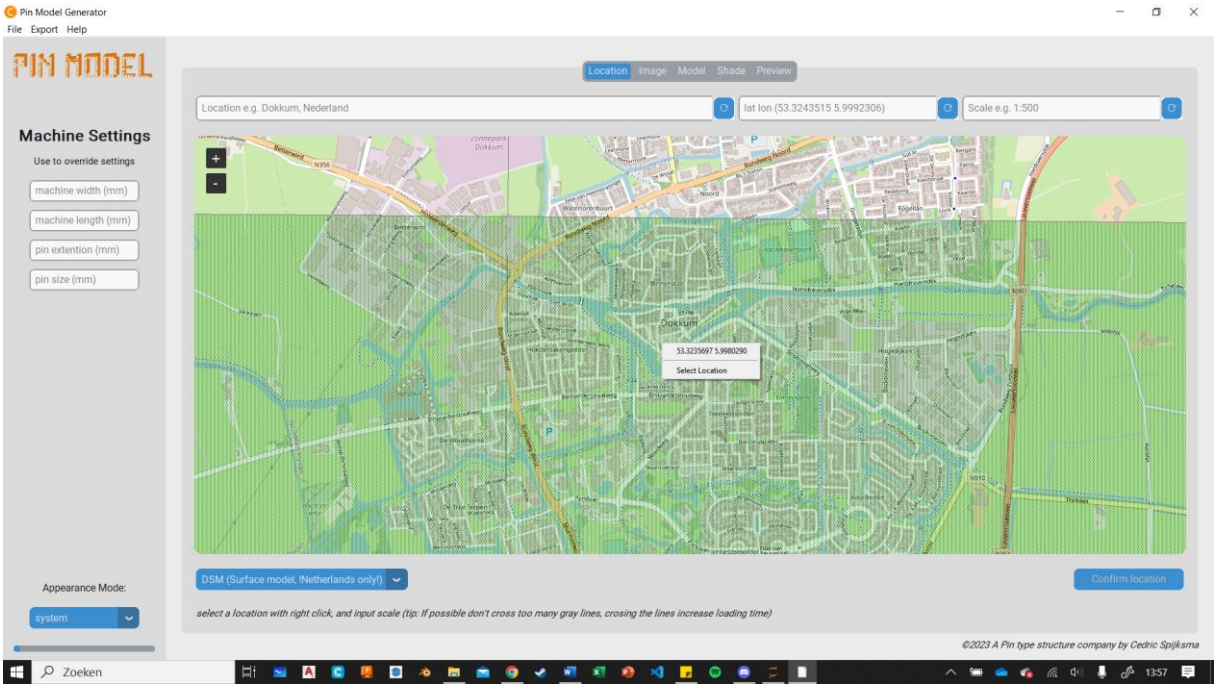
Openen settings.py and change the settings to match your machine.
the most important settings to check are: machine_width, machine_length, pin_size, pin_extention, servos, Default_scale
The rest of the settings are relatively general and, in most cases, don't have to be changed.

```
"""
this file is used to set te default settings of the programme
please be carefull when editing this file, incorect values can break the programm.
leave the hashes(#) in place and ONLY edit the values after the equal(=). when text is used use (")
"""

version = "V2.4.6 -- Development"
Language = "EN"             #supported: EN,NL,DE,TR
window = f"{1280}x{720}"    #format of window when opening, format: f"{1280}x{720}"
appearance_mode = "System"  # "System", "Dark", "Light"
block_size = 256            #increase for faster computers
pointcloud_preview = 8192   #amount of points to preview
chunk_size = 1000000        #increase for faster computers

machine_width = 140         #width in mm (default: 841)
machine_length = 180        #length in mm (default: 1189)
pin_size = 4.2              #size in mm (default: 3.5)
pin_extension = 40          #extention in mm (default: 180)
xy_speed = 100              #movement speed in mm/s (default: 100)
servo_speed = 0.15          #movement speed in sec/60° (default: 0.15)
servo_rotation = 180        #in degrees (default: 180)
servos = 2                  #up to 40 (see documentation for more info)
servo_gear_diameter = 62    #diameter in mm
gap_distance = 20           #distance between the cable on the z-gantry and the metalrods in mm

default_scale = 500         #default model scale(default: 200)

image_dpi = 300             #Image DPI (default: 300)
image_size= (16,9)          #Image DPI (default: (16,9))
```

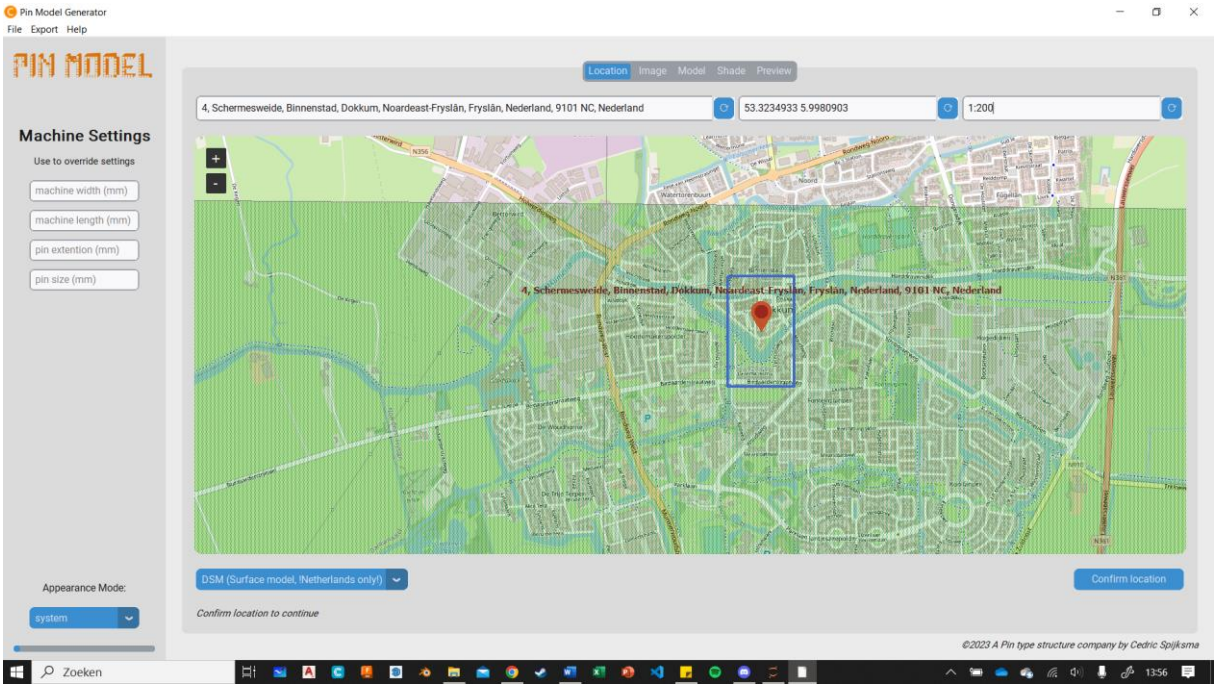
Step 2: Open the program



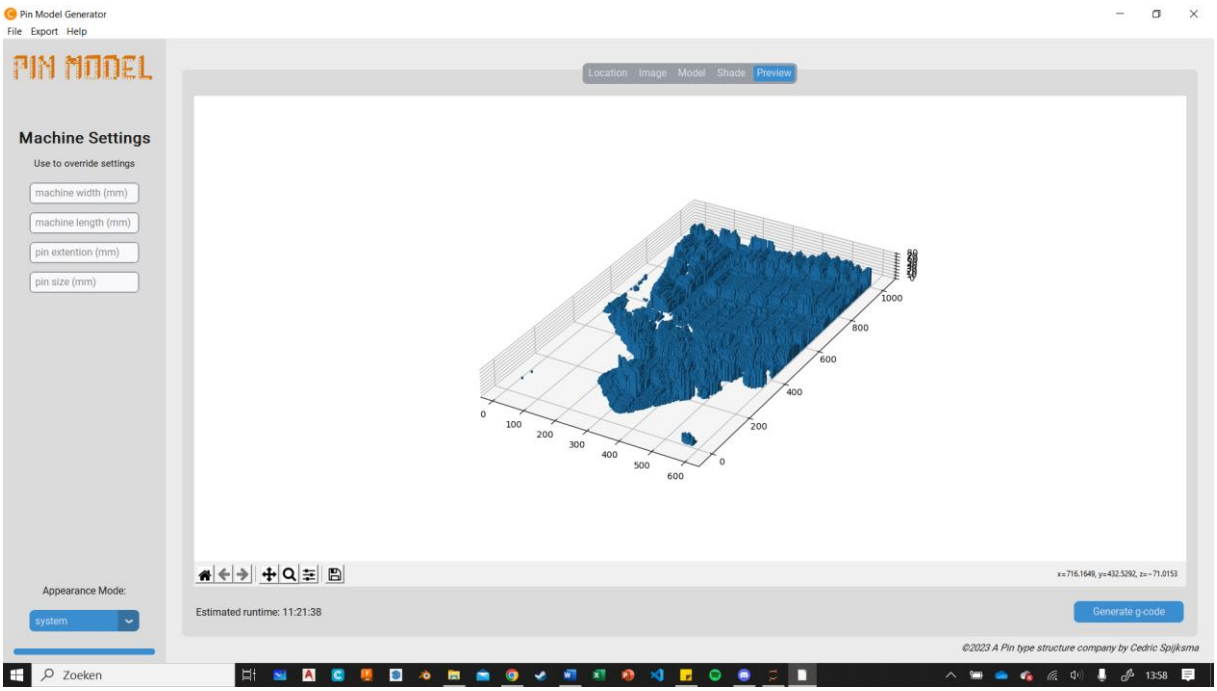
Step 3: Select a location on the map using right-click, and select “select location”



Step 4: Check the scale and see if the selection is correct. Press “confirm location.”
If the maps are not downloaded yet, a popup will appear. Click “Yes” to continue.
The program will now run. This can take up to several minutes if the maps are not downloaded.



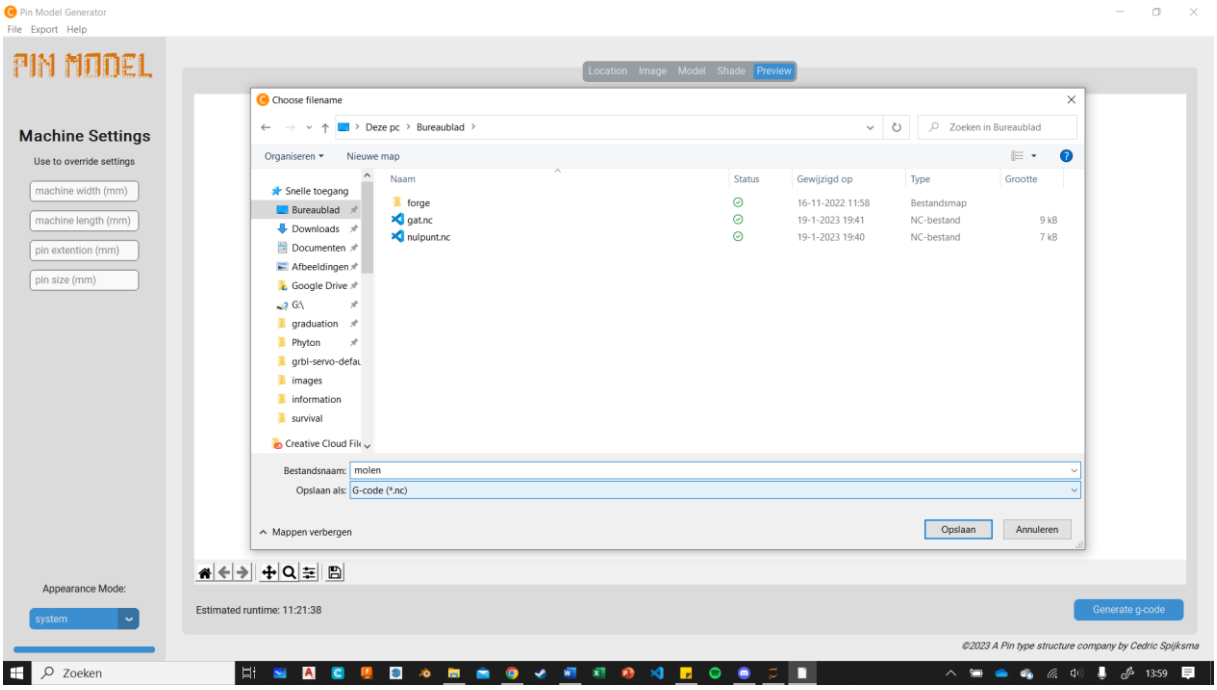
Step 5: check the result.
A preview will be shown based on the inputted data. When you are happy with the result, press generate g-code



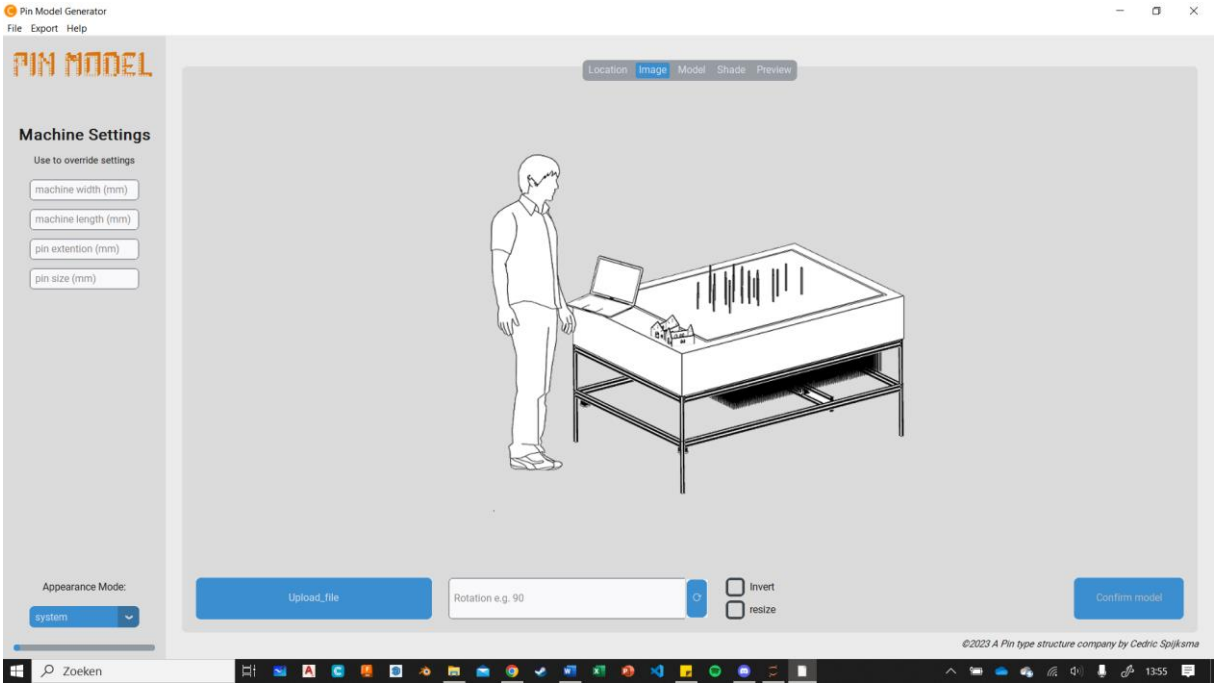
Step 6: save g-code

When you press, generate g-code, a pop-up window will appear. Here you can give the file a name and save it in a convenient location.

Go to the manual uploading file to continue.



Step 1b: uploading an image



step 1c: uploading 3D file