TB-RAM CoE – Technical documentation

Current and previous project workers: Rayko Toshev, Mika Billing, Miguel Zamora. Cordero, Dennis Bengs, Ricard Bitria Ribes, Tharanath, Ibukun Odubogun, Bart Klaster
Written by: Miguel Zamora. Cordero, Dennis Bengs, Ricard Bitria Ribes

Abstract

The purpose of this documentation is to:

- Explain the idea behind Robotic Additive Manufacturing.
- Describe the design process of improving on an existing RAM platform.
- Describe the interoperability between classical Additive Manufacturing and Robotic Additive Manufacturing.
- Describe a process of freeform additive manufacturing.
- Showcase the results of both regular and freeform Robotic Additive Manufacturing.
- Describe how to assemble, calibrate and run the Robotic Additive Manufacturing platform.

Abbreviations

3D	Three Dimensional
ABS	Acrylonitrile butadiene styrene
AM	Additive Manufacturing
RAM	Robotic Additive Manufacturing
CPE	Chlorinated polyethylene
DOF	Degrees of freedom
FDM	Fused Deposition Modelling
IK	Inverse Kinematics
PLA	Polylactic acid
PP	Polypropylene

Glossary

Term	Explanation
Additive manufacturing	3D printing or additive manufacturing is the production of real objects from 3D models. This is usually accomplished by extruding thermoplastics layer-by-layer on a print surface.
Build platform	The build platform is the plate or object on which the printed object sticks. The surface of a build platform come in many different materials, and the platform itself can include a heater to prevent the printed object from warping and unsticking. The build platform is also called a build plate, print bed or simply bed.
Degrees of freedom	In the context of robotic arms, degrees of freedom are the number of ways the arm can move.
End-effector	A device or tool connected to the end of a robotic arm. Examples of such devices are grippers and welders.
Extruder	The extruder is the part of a 3D printer that feed the filament towards the hot end. The extruder is commonly referred to as the “cold end”. The extruder commonly uses a stepper motor connected to a filament drive gear.
Filament	In the context of additive manufacturing/3D printing, filaments are thermoplastics shaped into thin long strands. Filaments are usually sold as rolls by weight.
Freeform printing	Freeform printing is a form of additive manufacturing in which the material can be deposited at multiple positions and angles in three dimensions, as opposed to layer-by-layer AM which is restricted to horizontal layers.
Fused deposition modelling	Fused deposition modelling or fused filament fabrication is a 3D printing process in which a continuous strand of filament is pulled through an extruder into a heated chamber and deposited on a growing work.
Inverse kinematics	Inverse kinematics is the mathematical process of calculating the joint parameters of a skeleton/armature to put its end-effector at a target position and orientation.
Layer-by-layer	Layer-by-layer is the additive manufacturing process in which a physical object is built up by flat horizontal layers. This is the most common form of additive manufacturing.
Hotend	The hotend is the part of a 3D printer that melts and extrudes filament. The hotend consists of a heating chamber and nozzle.
Robot/Robotic arm	A programmable mechanical arm or arm-like machine commonly equipped with an end effector and used to perform repetitive work.

Robot singularity	A robot singularity is a robot configuration in which the inverse kinematics breaks down. Singularities can cause unexpected very rapid movements.
Slicer	A slicer is a computer programs which converts a 3D model into 3D printer instructions, typically in the form of G-Code files.
Thermoplastic	Thermoplastics are plastic materials that become pliable when heated and solid when cooled.
G-Code	A programming language often used with CNC machines and 3D printers.

Table of contents

Introduction

Additive Manufacturing, also known as 3D printing, is a technology which has grown popular for doing rapid prototyping of objects. Using additive manufacturing, digital models can be turned into real objects made from a variety of materials.

Additive manufacturing is not just available for industrial applications. The price of 3D printers has fallen significantly over the last decade, enabling smaller companies as well as hobbyists to take advantage of the growing technology.

The most common type of additive manufacturing uses layer-by-layer plastic deposition to construct objects. Robotic Additive Manufacturing (RAM) extends the classical approach of 3D printing by mounting a 3D printer on a robotic arm, overcoming the physical limitations of standard 3D printers. This gives RAM the ability to not only do layer-by-layer deposition, but to also do freeform and large-scale prints.

During the TechnoBothnia Robotic Additive Manufacturing Center of Excellence project (TB-RAM CoE) a previous RAM platform was further developed and improved upon. New hardware was designed to allow for better control of the RAM process and software was developed to take better advantage of the robotic arm. This includes support for freeform printing and automatic bed leveling.

The technical documentation is split up into five chapters:

- Chapter 1 covers information about the.
- Chapter 2 covers the development of the physical hardware.
- Chapter 3 covers the development of the software packages.
- Chapter 4 covers the assembly of the extruder tool.
- Chapter 5 covers the setup and usage of the RAM platform.
- Chapter 6 showcases objects manufactured using the RAM platform.

Chapter 1

About the robot

This chapter is a brief introduction into the robotic arm used within the project.

1.1 What are robotic arms?

Robotic arms are mechanical arms used in many industrial applications, such as welding, painting, drilling and material handling. Robots excel at performing repetitive tasks inside a controlled environment, and without human supervision. This makes robots suitable for production line work.

Robotic arms generally carry an end-effector, which are peripheral devices attached to the robot's wrist to perform certain tasks. These tasks include welding, grabbing, painting, probing and now even for additive manufacturing. In this documentation a custom end-effector is described used for additive manufacturing. End-effectors may also be called tools or End-of-Arm-Tooling (EOAT).

Robotic arms come in many forms. Some robotic arms are human-like, with a shoulder, elbow and wrist joint. Other arms show little resemblance to any biological counterpart, such as SCARA robots.

A significant difference between robotic arms is the number of joints between their base mount and end-effector. The number of joints corresponds to a robot's Degrees of Freedom (DOF). The more degrees of freedom a robot has the more flexible it is within its own workspace, allowing the robot to manipulate objects from further away and from multiple angles. At least six degrees of freedom are needed to allow the end-effector to reach an arbitrary pose inside the workspace. The six-axis configuration is one of the most common setups found in welding robots.

1.2 ABB IRB-1200 90/5

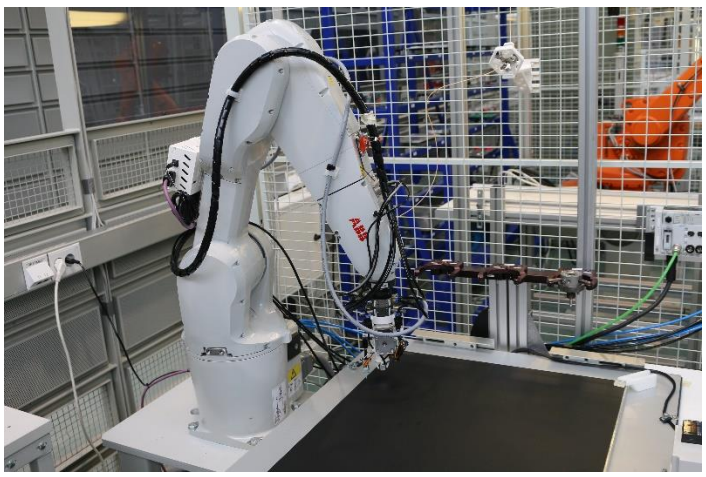


Figure 1: *IRB-1200*

The articulated robotic arm used in the TB-RAM CoE project is the IRB-1200 90/5 from ABB (see figure 1). The IRB-1200 is a six-axis industrial robot. The arm follows the classical six-axis joint configuration, with a base, shoulder, arm, elbow, forearm and wrist joint (names may differ).

The end-effector is attached to a pneumatic tool flange. The maximum payload of the end-effector is 5 kilograms.

The robot has several connection points for both pneumatic tubing and for power and/or data communication.

The robot program commonly used for the IRB-1200 is ABB's own simulation and programming software RobotStudio.

1.3 IRC5 robot system

The IRB-1200 uses a separate controller cabinet called IRC5, which connects to and controls the robot from a few meters away.

The IRC5 cabinet uses a FlexPendant. The FlexPendant is handheld device which allows full control over the robot. Among other things, the FlexPendant can be used to manually jog the robot and to start and stop robot tasks. The FlexPendant has a joystick built-in for fine control of the robot's motion.

Within the project, the FlexPendant is used to start and stop prints, to perform bed leveling and other utility tasks and to keep track of the print progress.

Chapter 2

Hardware development

During the project, several custom hardware components were designed and created to aid in the goal of robotic additive manufacturing. These include components like robot extruder end-effectors, air cooling systems, mainboard housing and filament holders. Most of these components were designed using Siemens NX 12. The prototype components and final designs were in PLA using Ultimaker. The following chapter will detail these components.

2.1 Extruder

For doing robotic additive manufacturing, a thermoplastic extruder tool end-effector was developed based on designs from previous projects involving RAM. This extruder tool acts as an end-effector for the IRB-1200 robot (see figure 2.1). The extruder tool is controlled by a separate MKS Gen 1.4 mainboard attached to the robot controller using a RS-232 connection.

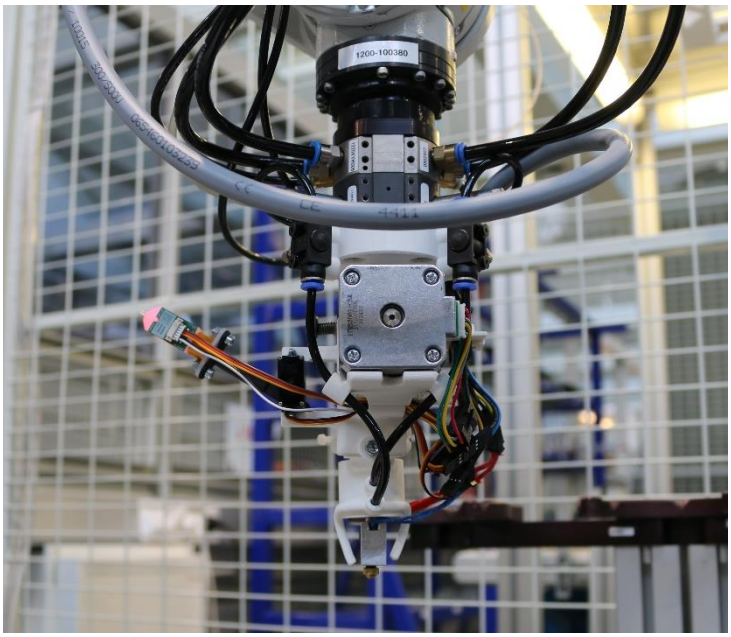


Figure 2.1: *Extruder tool*

2.1.1 Extruder mainboard and mainboard housing

The extruder tool is controlled by a separate MKS Gen 1.4 mainboard (See figure 2.2). Several other commercial mainboards were considered. The MKS Gen 1.4 was picked due to several reasons:

- Popularity and choice: The MKS Gen 1.4 is a very popular board focused on 3D printing. Due to its popularity, it is easy to find and order online.
- Price: Replacement boards are cheap if anything breaks.
- Large community: The MKS Gen 1.4 has a large community behind it which aids in troubleshooting and developing for the board.

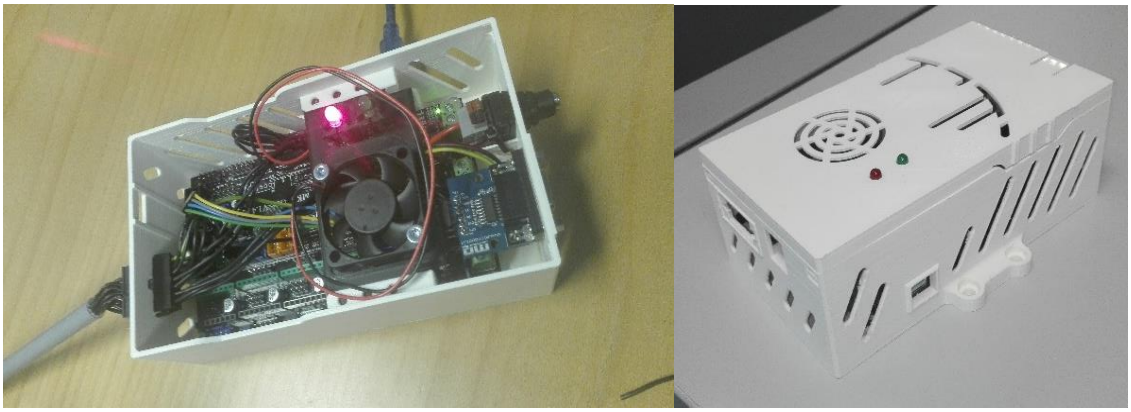


Figure 2.2: *Mainboard and housing*

To keep the mainboard secure and out of view, a custom housing was designed. The housing has several holes for external connections such as power and RS-232. The housing has an easily removable lid and cooling fan/status light base. Below the housing is a holder for a power supply unit. The housing uses snap-fit components for all detachable parts. The housing is attached to the back of the IRB-1200 robot using mounting screws.

For more information about the assembly of the extruder mainboard housing, see chapter 4.3.

2.1.2 Extruder tool

The extruder tool is an end-effector for the IRB-1200 robotic arm. The extruder tool serves multiple purposes:

- It pulls in the thermoplastic filament using the extruder motor and gear.
- It heats up the hotend which melts and extrudes the thermoplastic.
- It keeps track of the temperature of the hotend.
- It deploys and stows the bed leveling probe.
- It uses air cooling to dissipate heat from the hotend and printed parts.

The chassis of the extruder tool passed multiple iterations before arriving on the final design. The original design of the extruder tool had severe limitations which made is suboptimal for RAM. For example, the original extruder tool was very wide, making freeform printing difficult. In addition, the original tool relied on nuts and bolts to keep the tool aligned.

The new extruder tool is designed to be a single piece between the robot tool flange and the BondTech extruder. The original extruder tool relied on DC fans for air cooling. The new tool uses compressed air supplied through the robot.

The extruder tool has many electric components which are controlled by the extruder mainboard. Many different types of connectors have been considered for ease-of-use and power requirements (See figure 2.3).

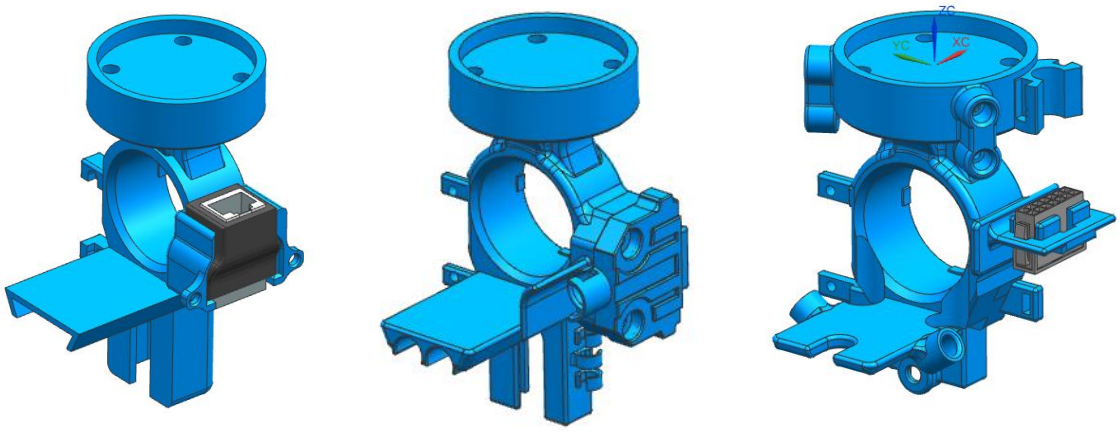


Figure 2.3: *From left to right: Network connector, Serial connector, Micro-Fit connector*

For more information about the assembly of the extruder tool, see chapter 4.2.

2.1.3 Air vent

An air vent component was designed to supply compressed air through the robot to the hotend heat sink and to the part vent independently (see figure 2.4). Several considerations were made when designing the component. The size of the component should be minimized as to allow for better versatility of the robotic arm during freeform printing. The air vent is detachable from the main extruder tool, as to allow for easy replacement and upgrades.

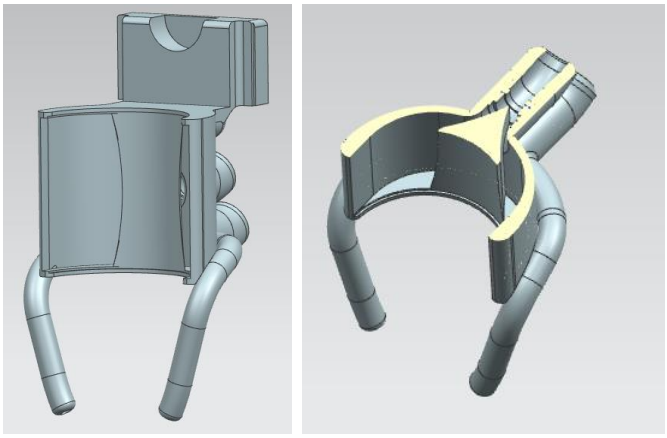


Figure 2.4: *External and Internal design of the Extruder tool air cooler*

The air vent includes two 4 mm air channels for tubes connecting to two air regulators valves which in turn connects to the robot. These valves allow for adjusting the air flow when the robot is stopped. Simulations of the air flow were created in ANSYS to check air flow behavior at 4.5 bars (see figure 2.5).

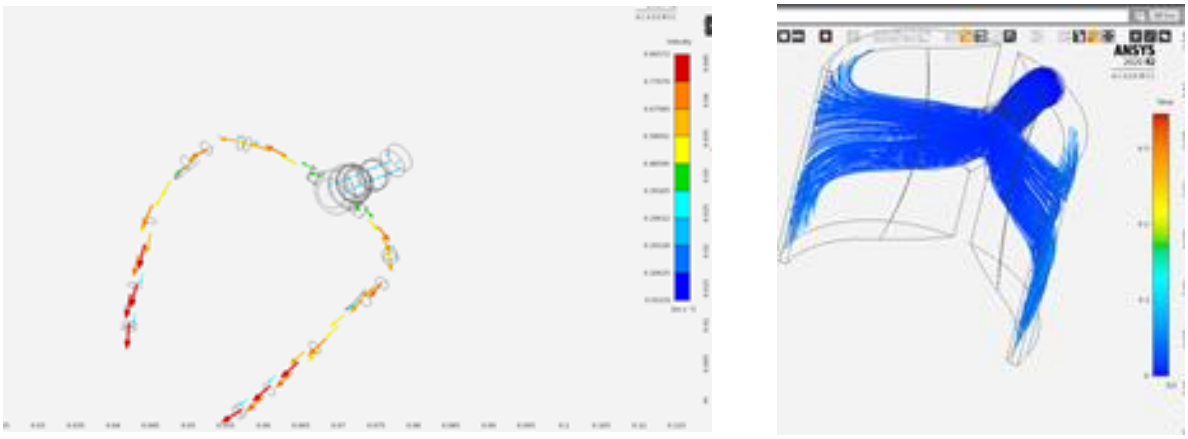


Figure 2.5: *ANSYS simulation in the air*

2.1.4 Bed levelling probe

To get an optimal first layer during additive manufacturing, the distance between the hotend and the platform must remain constant. If the distance fluctuates, the first layer will get an uneven surface. If the distance is off by a large amount, the hotend may get damaged when printing the first layer.

When using the previous extruder tools, the process of calibrating the distance between the hotend and build platform were manual and limited to a 3-point definition. In this case, any irregularities in the build platform itself would case the first layer to be uneven. To solve this issue, an automatic bed leveling probe was integrated into the new extruder tool design.

The probe allows automated leveling of the build platform. A digital servo based on the MG90 form-factor controls the deployment and stowing of a BLTouch 3D printer probe (see figure 2.6). This ensures that the probe stays out of the way during the printing process, which aids in freeform printing.

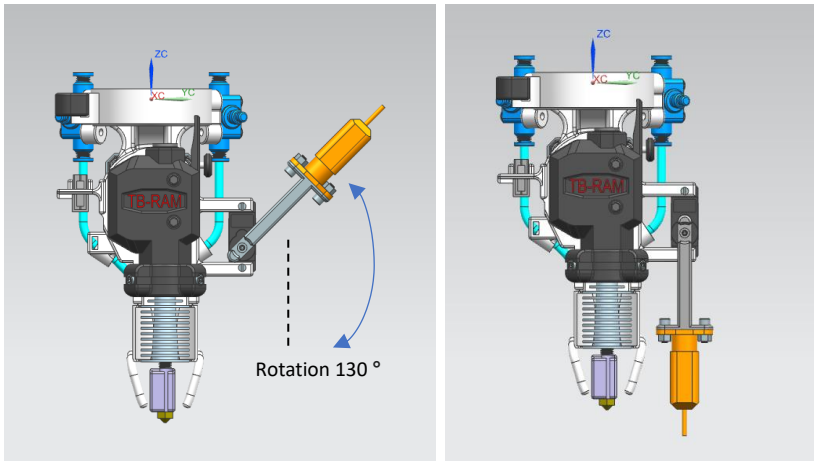


Figure 2.6: *Motion of the bed levelling probe*

2.1.5 Heat-Sink holder

The hotend part of the extruder tool relies on a heat sink to keep the hotend at an even temperature and to prevent the extruder itself from melting. The hotend is made from a heater block attached to the heat sink through a heat break pipe.

A heat sink holder was designed to hold the hotend assembly in place from the extruder tool chassis. The holder incorporates a spring mechanism which allows the heat sink to be pushed inwards a short distance. The idea behind this mechanism is to prevent damage to the extruder tool and/or robot in case of small platform collisions.

Several versions of the spring mechanism were designed and tested. One issue encountered with the spring mechanism is that it did allow for a small amount of horizontal movement of the hotend assembly. Another issue was that the vertical movement had too much friction, leading to the spring mechanism not returning to its original position when depressed.

For more information about the assembly of the heat sink holder, see chapter 4.2.

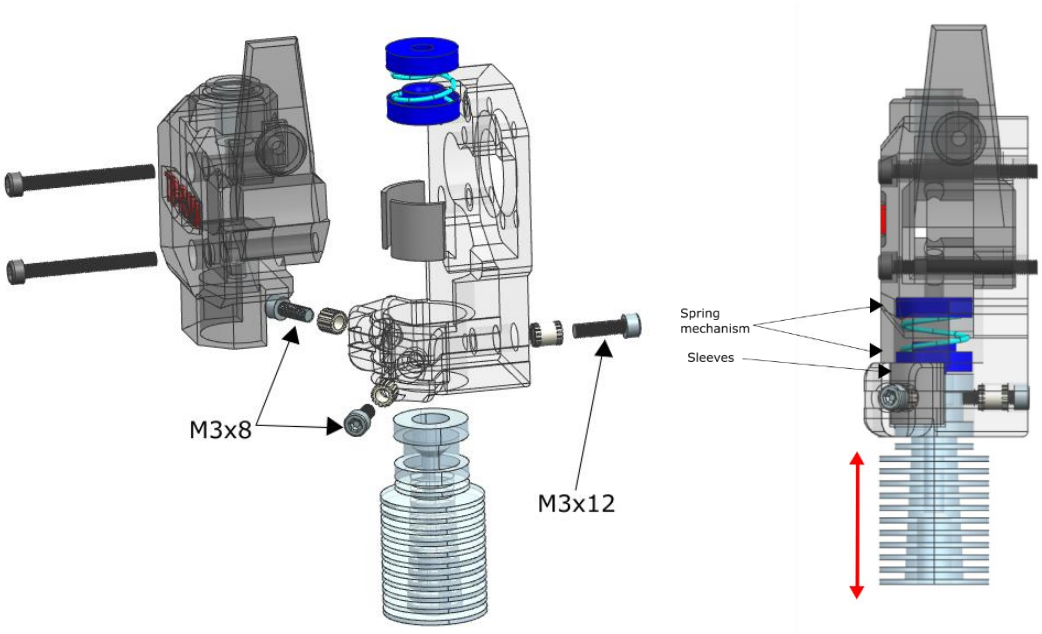


Figure 2.8: *Assembly and motion of the heat sink holder*

2.2 Build platform

Multiple build platforms were developed and used for RAM. Version 1 and 2 of the build platforms were unheated. The third and final version uses several thermostat-controlled heaters below a large aluminum block which keeps a glass bed heated during the printing.

The print surface of all the platforms was BuildTak, a special type of sticky plastic coating with good adherence with many thermoplastics. There are exceptions such as when printing Polypropylene, which required an additional coating to stick.

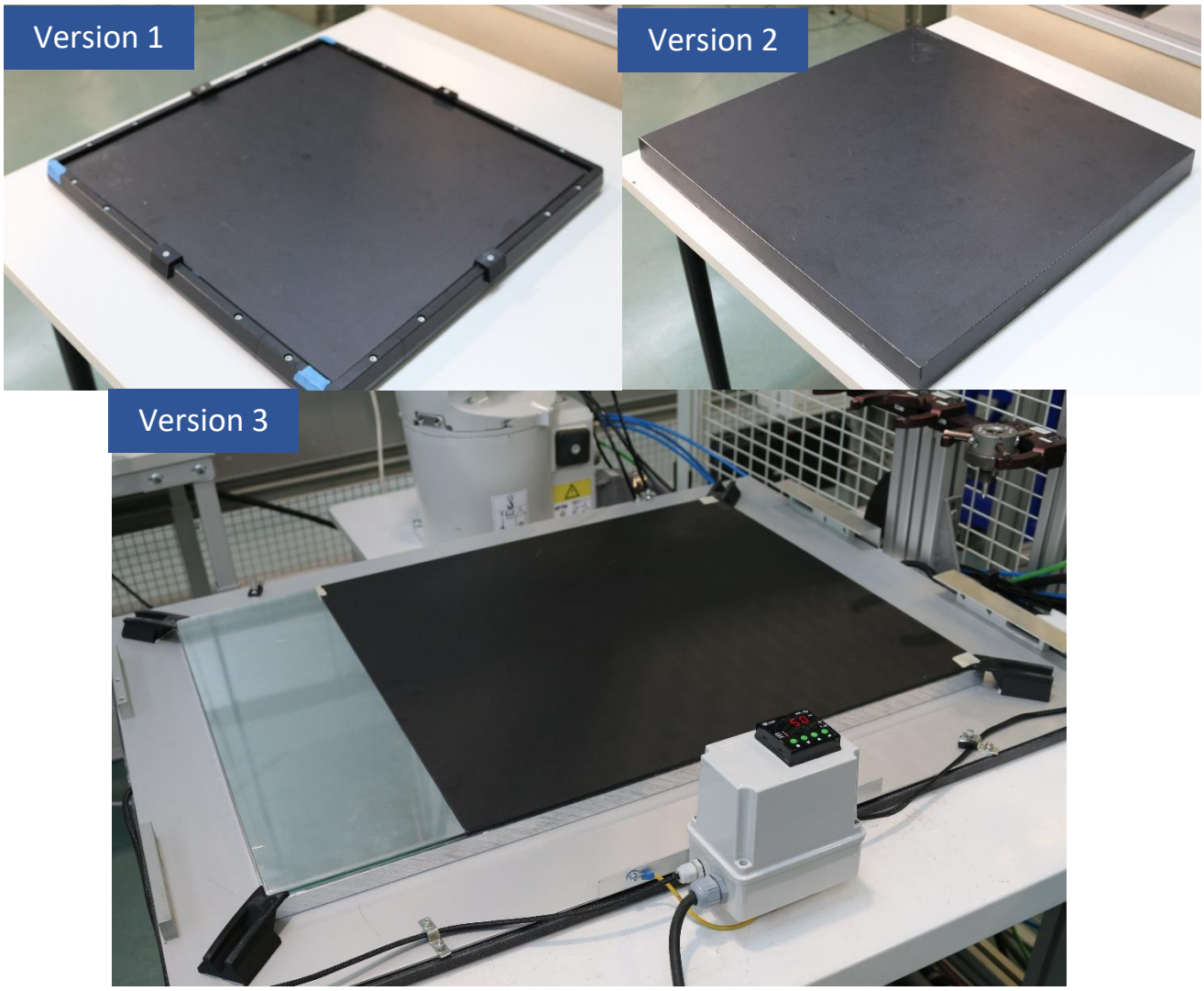


Figure 2.7: *Build platform versions*

2.3 Other parts

Several other components were designed for other purposes within RAM. These include but are not limited to:

- a filament spool holder (see figure 2.9 and figure 2.11): The spool holder holds the filament roll in place during printing.
- a filament guide (see figure 2.10 and figure 2.11): The filament guide limits the movement of the filament strand exiting the filament spool and keeps the filament spool from unrolling itself. The filament guide uses ball-bearings to smoothly guide the filament strand through a center hole.
- build platform holders: These keep the glass plate in the build platform from moving.

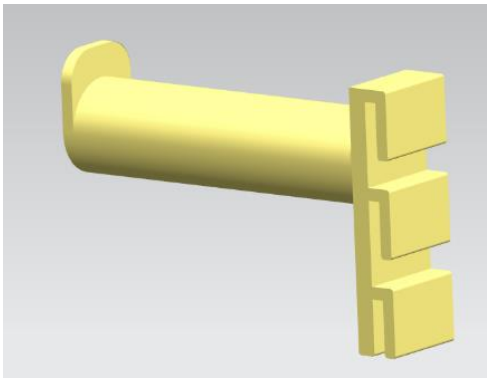


Figure 2.9: *Spool holder*

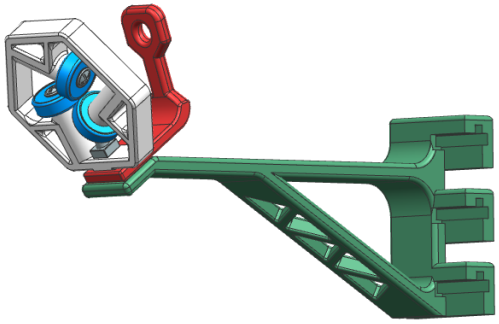


Figure 2.10: *Filament guide*

Figure 2.11: Assembled spool holder and filament

Chapter 3

Software development

Several pieces of software have been developed within the project to facilitate both regular and freeform additive manufacturing using the robotic arm. This includes slicing software, microcontroller firmware and robot program.

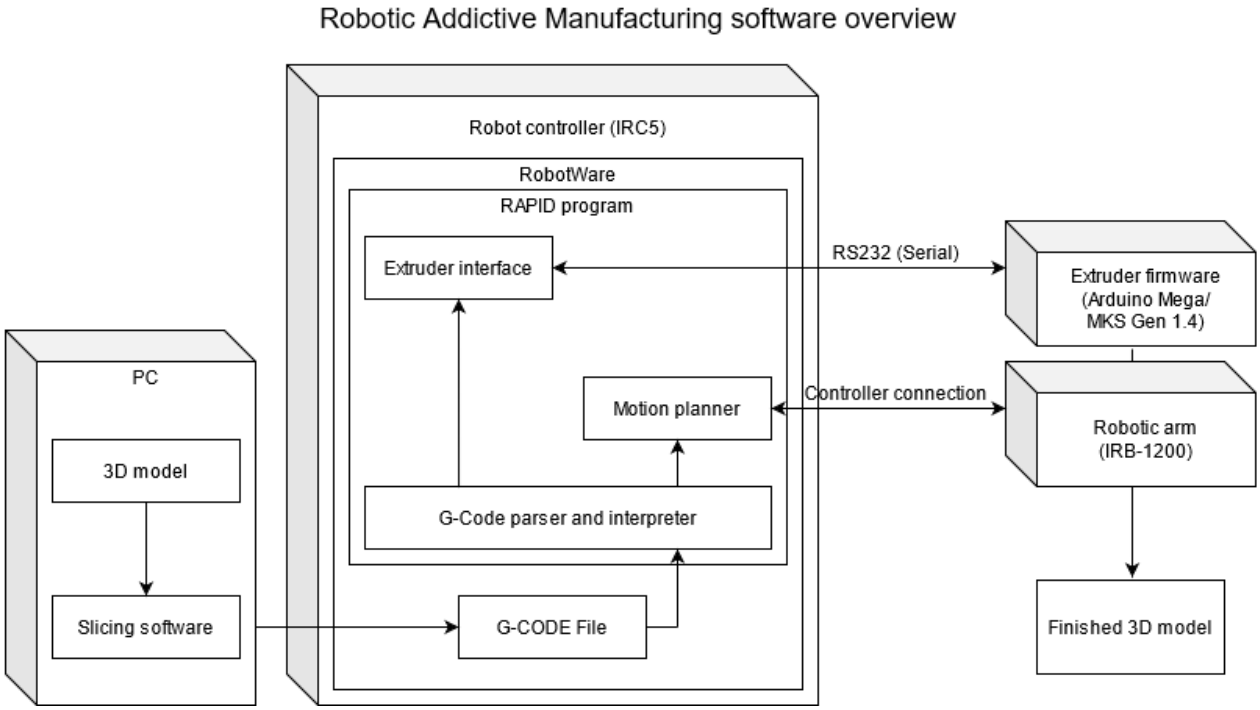


Figure 3.1: *RAM software overview*

3.1 G-Code

G-Code (Geometric Code) is a programming language which is commonly used with CNC machines. G-Code has come to be the de-facto standard for 3D printers.

G-Code files usually contain a long list of G-Code commands (or blocks) which tell a machine what to do to complete a certain process. In the case of 3D printers, this process is typically to:

- Set heater and bed temperature.
- Set fan speed.
- Move hotend.
- Extrude plastic layer by layer.
- Gracefully stop.

These actions are accomplished by small G-Code commands which perform simple steps. The most common G-Code commands are G0 and G1. These commands tell the machine to move one or more axes to a target position at the given speed. For a full list of G-Code commands used on 3D printers, see: https://reprap.org/wiki/G-code

While G-Code is mostly standardized, G-Code will not be the same for all machines. Each machine can have special commands and interpret G-Code commands in a different way compared to other machines. In the case of 3D printers, the difference in G-Code between machines is minor, but care must still be taken to avoid running incompatible G-Code files on the wrong machine which may damage the machine.

3.2 Slicing software

In additive manufacturing, the slicing software is a program which take a 3D model, and outputs the instructions for turning it into a physical object through a process called additive manufacturing. Most slicing software works on the principle of layer-by-layer printing and will generate instructions on a per-layer basis.

Slicers typically have many tunable parameters and features. These aid in finetuning and debugging the final quality of the printed object, as well in aiding in finding the optimal alignment and orientation to print the object or objects in.

The slicing software typically stores settings for different machines and filaments in slicing profiles. These profiles define the relevant settings required for that machine and filament, such as print bed dimensions, heater temperature, cooling fan speed, layer height and printing speed.

In addition, slicing software typically features:

- Infill pattern and volume: The infill is a pattern printed inside an object to both make it tougher and to give something for the top layers to adhere to. Modern slicers support many complex infill patterns, with each pattern having its own strengths and weaknesses. An example pattern from nature is the hexagonal infill pattern, which fills the object with a beehive-like pattern.
- Supports: Since layer-by-layer deposition cannot print structures freely hanging in the air, support structures are required where there are large overhangs. These support structures can be made in the same material as the original object, or they can be made in special materials which are easier to break away or dissolve.
- Rafts and brims: To further improve bed adhesion between the object and the build platform, a raft or brim can be added under or around the object.

For more information about the slicers used within the project, see chapter 6.2.

3.3 Robot software

The following chapter will explain the additive manufacturing robot program. The robot program acts as the main controller within the additive manufacturing process.

3.3.1 ABB RobotStudio

The IRB-1200 robot used in the project is programmed through ABB RobotStudio (see figure 3.2).

RobotStudio is both a simulation and a programming software developed by ABB. The program features an Integrated Development Environment for developing robot program. In addition, RobotStudio can simulate compatible robots inside a three-dimensional environment, allowing the user to safely simulate the developed software from inside a virtual environment.

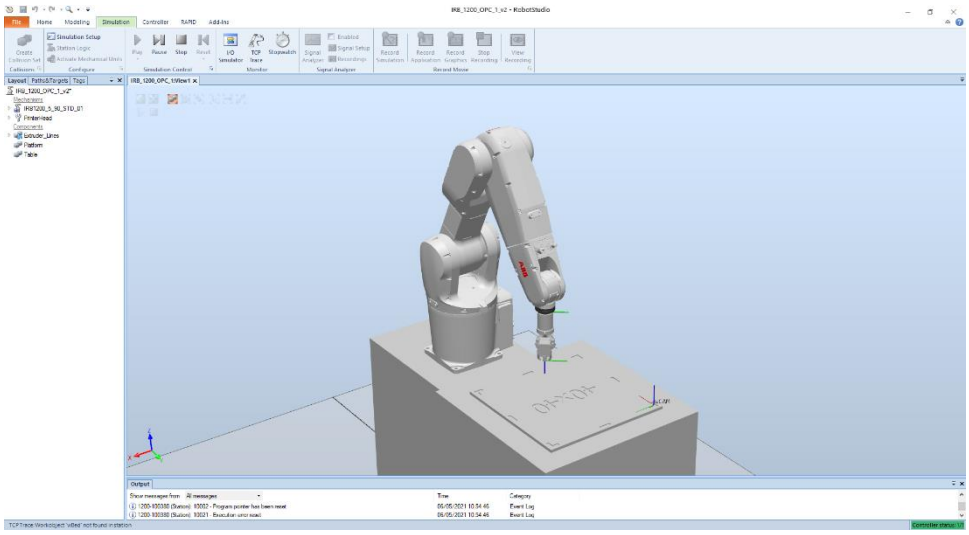


Figure 3.2: *Robotstudio virtual controller and robot*

RobotStudio can also connect to a real robot controller to allow the developer to upload and test the developed AM robot program on a real robot. In addition, the real robot can be visualized inside RobotStudio.

3.3.2 RAPID

RAPID is a high-level programming language which runs on top of the robot controller software RobotWare. RobotWare is a family of robot controller software.

RAPID has many features found in typical high-level programming languages. RAPID features procedures, functions, traps (error handling), standard arithmetic and multitasking.

RAPID has many built-in data types and functions specifically used to control robots. Common examples of these are:

```
tooldata    Definition of a tool (tool center point, weight etc.).
robtarget   A position and orientation in space.
MoveL       Command the robot to move linearly.
MoveJ       Command the robot to move by joint movement.
```

The robot program used in the project was written in RAPID.

3.3.3 AM robot program: RoboP3D

For controlling the robotic arm during the additive manufacturing process, a robot program nicknamed RoboP3D is used. The program was developed in-house for use within the TB-RAM CoE project.

The basic principle of RoboP3D is to translate G-Code commands into robot and extruder movements, as well as keeping track of the current state of the system. The program features many improvements over the original robot program developed in earlier projects. The improvements include but are not limited to:

- A fully featured G-Code parser (for more info on G-Code, see chapter 3.1).
  Most of the common G-Code commands used in additive manufacturing are handled gracefully. If an unknown G-Code command is encountered, the program can be configured to stop.
- Variable extrusion speed
  Like with other 3D printing machines, the extrusion speed can be adjusted through G-Code.
- Mesh bed levelling
  The program generates a bed levelling mesh by probing the height of the build platform at the specified resolution. Bilinear interpolation is used to adjust the height of the first layers during print to match the platform profile. This ensures an even connection between the printed object and the build platform (see chapter 3.3.4).
- Verbose error handling
- Freeform additive manufacturing
  The tooltip can be rotated to arbitrary angles using the rotational G-Code parameter A and B. This allows the program to do freeform additive manufacturing when used with compatible G-Code data (see chapter 3.5.1).
- Automatic stop and resume print functionality using a supervisor task.

RoboP3D is split into multiple modules. The names and functionalities of these are:

- `P3DBedCalibration`
  Handles bed definition and levelling.
  Can also be used to generate a 3D model of bed mesh.
- `P3DError`
  Handles errors from all modules.
- `P3DExtruderInterface`
  Handles all communication with the extruder.
  The module also keeps track of the current state of the extruder.
- `P3DExtruderTests`
  Runs a series of extruder tests.
- `P3DGCodeInterpreter`
  Parses and executes G-Code files, line by line.
- `P3DPersistent`
  Contains persistent variables. These variables must not be cleared on controller restart.
- `P3DPlanner`
  Handles all arm movements and path planning.
  The module also keeps track of the current state of the virtual 3D printer.
- `P3DProfile`
  Contains user-configurable parameters.
- `P3DUserMenu`
  Handles the main user menu.
  This is the main entry point to the program.
- `P3DWrite`
  Handles FlexPendant write throttling, and message boxes.
- `P3DExtruderSupervisor`
  This task supervises the main robot program and retracts filament on program stop or error.

For additional instructions how to use the code, see chapter 5.

3.3.4 Automatic bed leveling

The AM robot program features automatic bed leveling using a touch probe built into the extruder tool. Bed leveling is an important step in getting an even first layer during additive manufacturing. With an uneven first layer, the printed object may not stick to the build platform, there may be under- or over-extrusion, and the hotend may get damaged if it collides with the platform.

When using bed mesh leveling, a regular two-dimensional height grid of variable resolution is probed across the given dimensions of the build platform. The height of each point in the grid is determined by descending and ascending the touch probe in a binary search pattern until the probe is sufficiently close to the build platform (see figure 3.3).

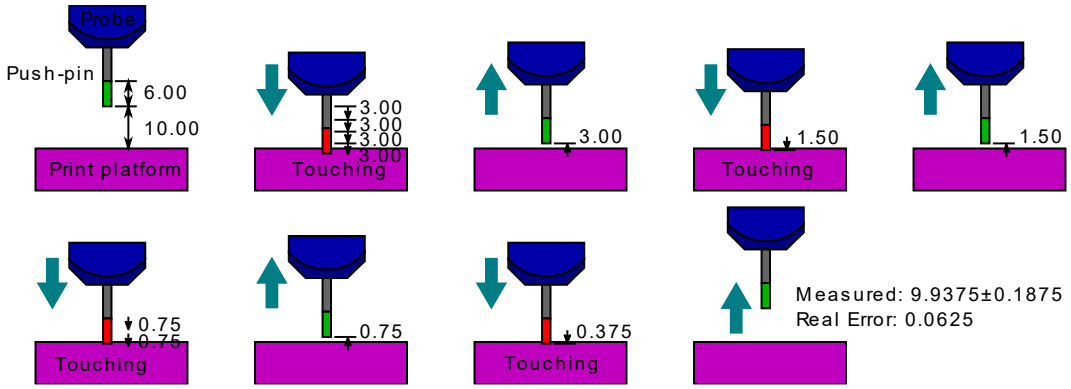


Figure 3.3: *Binary search bed leveling*

During the additive manufacturing process, the build platform height at each point below a given height is offset by a value determined by bilinearly interpolating the bed mesh leveling grid (see figure 3.4).

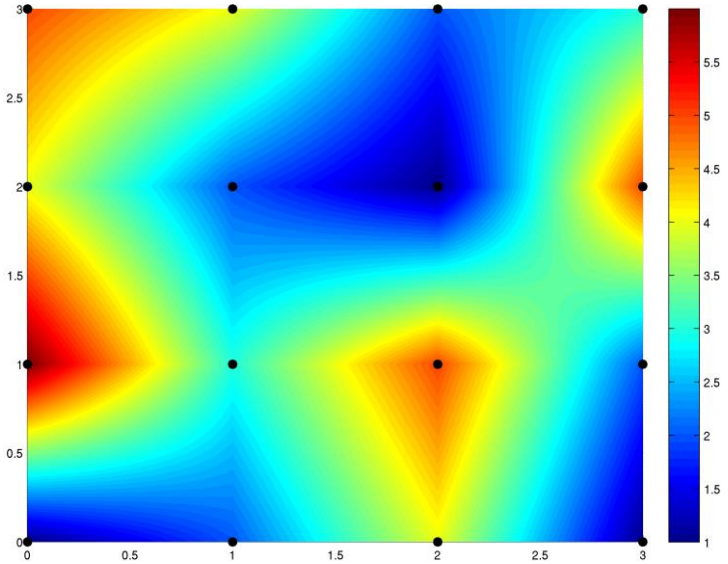


Figure 3.4: *Bilinear interpolation of a 4x4 grid*

This adjustment fades out over a set fade height. In practice this means that any irregularities in the build platform will not be visible in upper layers of the printed object.

3.3.5 G-Code compatibility issue

One of the inherent differences between a robotic arm and a standard 3D printer is that a 3D printer is optimized in executing many very small fine movements. Trying to execute the same code on a robotic arm with a much larger frame may cause certain issues.

These issues include:

- The robot may not stop in time (deceleration limit).
- The robot controller may not have time to process the next command in time and is forced to stop (corner path error, deceleration limit).

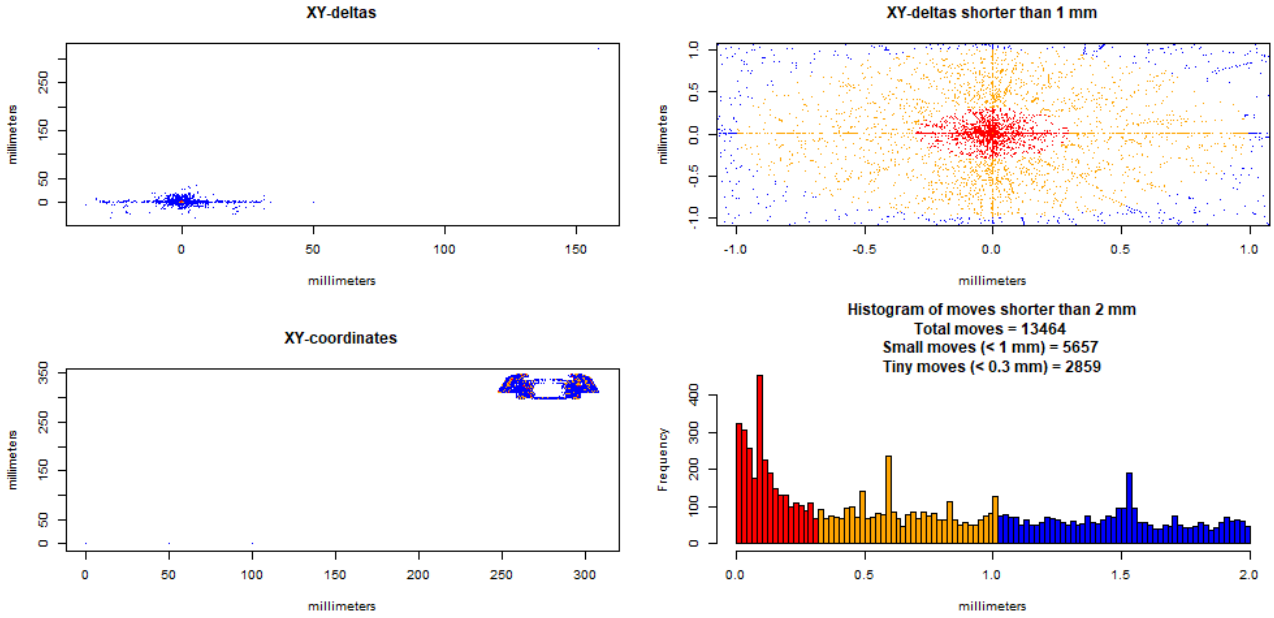


Figure 3.5: *Command statistics of a troublesome G-Code file*

In addition, a “Close to singularity” error may occur at certain spatial coordinates within the print area where the robot configuration approaches a singularity point.

Some of these errors are critical and will cause the print process to fail. An ad-hoc solution developed within the project was to use a supervisor task to gracefully pause the printing process in the case of robot error. This allows the printing to be resumed at a later point.

Another concern when translating G-Code to robot movements was that the absolute vertical stability may not be unison across the entire robot workspace. The precision of vertical movement may be lower at the far corners of the robot workspace, which may lead to the hotend colliding with the extruded print layers if the layer height is too low.

3.3.6 Synchronization issues

Due to the rapid movements used in additive manufacturing, and the requirement for an even flow of filament in a given distance, synchronization of the filament extruder and robotic arm movements is of critical importance.

Ideally the time it takes to move the robot end-effector from point *A* to point *B* should be deterministic, but due to real-world considerations such as acceleration and inertia, the duration of the move will not be predictable.

Due to the filament extruder being controlled by a third-party microcontroller, there is no direct communication between the robot hardware and microcontroller. The communication between robot controller and extruder microcontroller is done over an RS-232 connection through the robot controller's serial port.

While this communication bus alone introduces a small delay in communication between the robot and extruder, there is also a source of delay in the RAPID code execution itself.

To briefly summarize the issue. The RAPID code is executed using a certain prefetch time interval before the robotic arm reaches a target point. During this prefetch time interval, the RAPID code can calculate the next target point for the robot. This allows the robot to immediately continue onto the next point without stopping and waiting. The prefetch time is essential in additive manufacturing. Without it the printing process would be significantly slower as the robot is forced to stop and wait between each target point. The prefetch time however makes it difficult to synchronize the robot motion with the extruder motion, as the RAPID code is not executed at corner points, or in additive manufacturing, at the start and end of lines.

Many workarounds have been considered for this issue, from relying on digital output signals from the robot controller, to using accelerometers and other methods of synchronizing the extruder with the robot motion. The current solution in the robot program is to do a full stop of both the filament extruder and robotic arm at critical points, and restart both the extrusion and robot motion at the same, or as close to the same time as possible.

3.4 Extruder firmware: xstrudt

The extruder firmware, nicknamed xstrudt, is the program which controls the motion of the filament extruder motor as well as powering and regulating the hotend heater.

The firmware is written in the Arduino Integrated Development Environment in C/C++. The firmware controls a MKS Gen 1.4 microcontroller.

The microcontroller communicates with the robot controller over a RS-232 connection. The messages sent between the two systems are written in plain-text ASCII, and follow a protocol given in the file `rs-232_protocol.h`. The firmware makes use of several of the built-in timers on the board.

The extruder firmware is split into multiple header files. The files are:

- `xstrudt`
  Main entry point and loop.
  Also handles incoming serial commands, and heater temperature control.
- `conf.h`
  Includes constants used in the firmware.
- `fletcher16.h`
  Calculates Fletcher16 checksums used in serial communication.
  For more info, see: https://en.wikipedia.org/wiki/Fletcher's checksum.
- `probe.h`
  Control of the BLTouch servo and probe deployment servo.

- *pwm.h*
  Pulse-width modulates a single pin.
  Used to regulate heater temperature using a given duty cycle.
- *rs-232_protocol.h*
  A description of the communication protocol used between the robot controller and extruder microcontroller.
- serial.h
  Helper functions for serial printing.
- stepper.h
  Controls the filament stepper motor.
- thermistortable.h
  This file contains a list of thermistor voltages and temperatures. This list MUST be updated if a new type of thermistor is used with the hot end, or the temperatures will be wrong.

3.5 Freeform additive manufacturing

By utilizing the flexible nature of robotic arms, it is possible to enable full freeform printing when doing robotic additive manufacturing.

With freeform printing, the hotend can be positioned to and extrude material at most of the exposed surfaces of a 3D model. With regular FDM, only the topmost layer is considered a viable target for plastic deposition. Through the freeform printing process, complex objects can be created which would otherwise be impossible for standard 3D printers, such as negative overhang structures.

3.5.1 Freeform G-Code and slicer

Freeform printing was enabled by adding two additional G-Code parameters to the original robot program G-Code interpreter: *G0 & G1 A* and *B*. The *A* and *B* parameters define the hotend rotation about axis X and Y respectively.

Regular slicing software do not use these parameters, as they do not allow for freeform printing. A separate slicing software was thusly written to enable testing and debugging of the freeform printing capability of the robot program.

The freeform slicer program is a small program written in JavaScript, using the Three.js WebGL framework for model visualization.

The slicing program is used produce freeform G-Code from a parametric/mathematical model. The program cannot be considered a general slicing software, as it does not take a 3D model as input. Instead, it relies on the user understanding 3D mathematics and programming their own parametric 3D model using JavaScript (see figure 3.6).

When developing a model, a series of lines are given in spatial coordinates. Each line has a width, height and an orientation. To facilitate the non-planar nature of freeform additive manufacturing, the tool is capable of gradually adjusting the width and height of a line between endpoints. The slicer component handles the filament flow calculations, model visualization and other low-level details.

```
static createWaveVase(slicer, wireframe) {
    const v2 = new THREE.Vector2();
    const p = new THREE.Vector3();

    const parameters = slicer.getParameters();

    const initialLayerHeight = 0.4;
    const layerHeight = 0.3;
    const baseRadius = 20;
    const lineWidth = 0.5;
    const waveMagnitudeX = 4;
    const waveFrequencyX = 4;
    const waveOffsetX = 0.001;
    const waveMagnitudeZ = 2.5;
    const waveFrequencyZ = 10;
    const segments = 100;
    const layers = 70;
    const maxZ = layers * layerHeight + waveMagnitudeZ;

    const getZ = (si, zi) => {
        return Math.max(initialLayerHeight, zi * layerHeight +
            THREE.Math.lerp(0, waveMagnitudeZ / 2, zi / layers *
            Math.sin(si / segments * waveFrequencyZ * Math.PI * 2)));
    };

    for (let zi = 0; zi < layers; zi++) {
        slicer.nextLayer();
        for (let si = 0; si <= segments; si++) {
            const r = baseRadius + waveMagnitudeX * Math.sin((si / segments + zi * waveOffsetX) * waveFrequencyX * Math.PI * 2);
            const x = (parameters.originAtCenter ? 0 : parameters.printerSizeMm.x / 2) + Math.cos(si / segments * Math.PI * 2) * r;
            const y = (parameters.originAtCenter ? 0 : parameters.printerSizeMm.y / 2) + Math.sin(si / segments * Math.PI * 2) * r;
            const z0 = getZ(si, zi);
            const z1 = getZ(si, zi + 1);
            const lh = z1 - z0;
            if (si === 0 || lh === 0) {
                slicer.moveTo(p.set(x, y, z0));
            } else {
                slicer.lineTo(p.set(x, y, z0), v2.set(0, 0), lineWidth, z1 - z0, true);
            }
        }
    }
}
```

Figure 3.6: *Parametric curved wase code*

The created model is then visualized within a 3D environment (see figure 3.7). The model is made from the line segments that the robot is expected to move along while extruding.

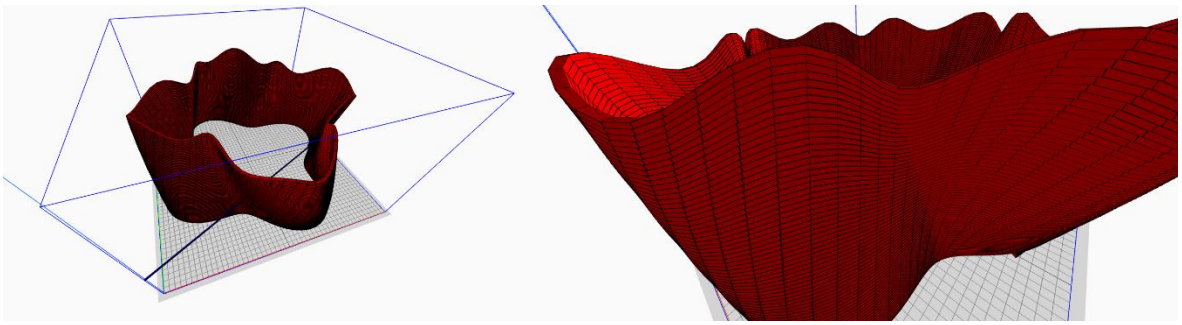


Figure 3.7: *Parametric curved wase model*

Using this software, multiple freeform models have been created, and several of these models have been successfully printed using the robot (see figure 3.8).

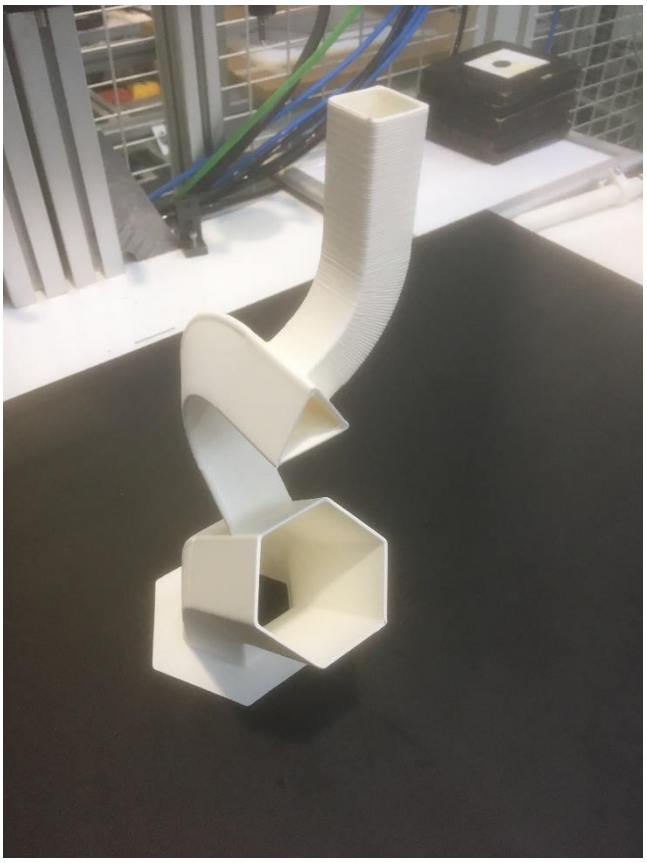


Figure 3.8: *Freeform nested pipes*

3.5.2 Freeform printing concerns and warnings

There are several potential dangers when doing freeform printing with a robotic arm. Before doing a freeform print, special care must be taken so that the print may proceed safely.

Unlike layer-by-layer additive manufacturing, freeform printing allows the robotic arm to bend freely around the model. This greatly increases the chance of expensive and hazardous collisions between the robotic arm and the build platform, the printed object, or the environment.

The goal of a robotic arm is to put its end-effector at the target position and orientation. This is usually done through a mathematical process called inverse kinematics. If the robotic arm has enough degrees of freedom, it is likely that there exists more than one way to reach the same position (see figure 3.9).

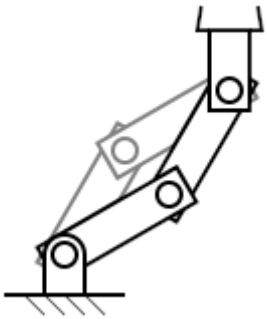


Figure 3.9: *Inverse kinematics chain with two solutions*

This is an issue as it makes the robot program less predictable. For example, one solution may be safe, while the other solution crashes the arm into a table.

This unpredictability can be alleviated by providing so-called axis configurations. These configurations force the robot joints into predetermined quadrants relative to each other.

Even so, in the AM robot program we have chosen to disable axis configurations and allow the arm to instead use the closest configuration automatically. This allows the robot to change configuration during the additive manufacturing process.

Care must be taken that the configuration changes do not twist, pull or otherwise damage the robot or the wires connected to the end effector. This is an issue which is likely to happen when passing through a singularity.

The following video has a short demonstration of the common singularities in a 6-DOF robot:

https://www.youtube.com/watch?v=lD2HQcxeNoA
What are robot singularities? by Mecademic Robotics.

Tips for avoiding issues:

- Run the G-Code in a virtual copy of the robot before the real robot.
- Avoid doing moves which bends the wrist joint across the center angle.

3.6 Other software

Various smaller utility programs and scripts have been developed under the course of the project.

3.6.1 Extruder interface

The extruder interface program was written in Python to facility debugging of the extruder firmware. It is also useful for calibrating the extruder and for changing filament The program has a simple window interface which allows full control over the extruder firmware (see figure 3.10). It communicates with the extruder over RS-232.

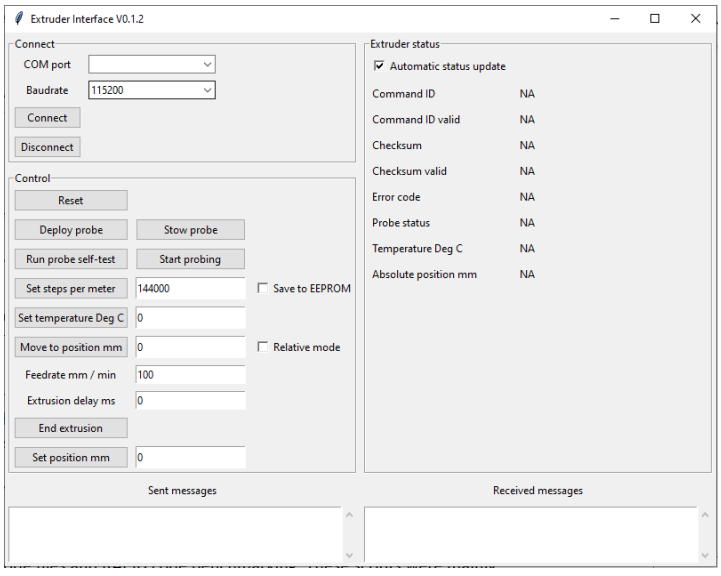


Figure 3.10: *Extruder interface*

3.6.2 R and python scripts

R is a statistical programming language with extensive plotting capability. Several scripts were written in R for analyzing G-Code files (see figure 3.11) and RAPID code benchmarking. These scripts were mainly used to analyze the compatibility of G-Code with the robot program. See chapter 3.3.5 for more details.

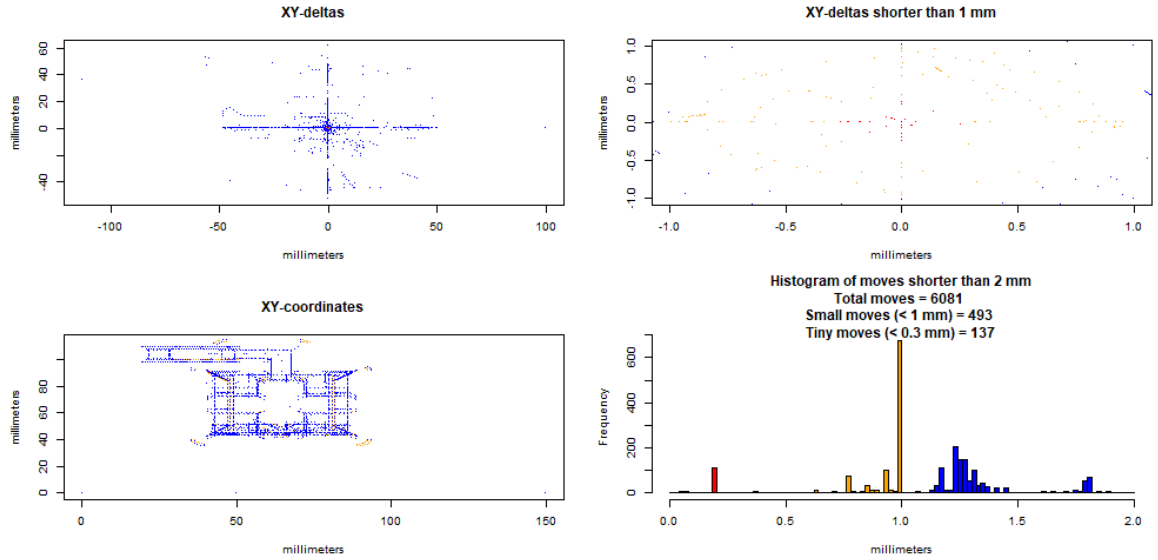


Figure 3.11: *Command statistics for Calicat print*

Python scripts were made for creating simple calibration prints for the robotic arm. These scripts were the precursor to the freeform slicer program, supporting very basic types of shapes.

Chapter 4

Extruder assembly

The following guide will in detail explain how to assemble the extruder mainboard and robot tool for the robotic additive manufacturing platform.

4.1 Part list

The following parts are required for the extruder tool and mainboard housing in addition to the printed parts:

#	Name	QTY
1	MKS Gen 1.4 mainboard	**1**
2	E3D V6 – all metal hotend (12V Direct drive, either 1.75mm or 2.86mm)	**1**
3	E3D Thermistor Cartridge (must be Semitec 104Gt)	**1**
4	Bondtech QR Extruder	**1**
5	MG90S servo (metal gears, digital)	**1**
6	BLTouch V2	**1**
7	Pneumatic Air Flow Control Valve Push-to-Connect Fitting 4mm	**2**
8	40 mm fan (12 Vdc)	**1**
9	Green LED (2.0-2.8 Vdc)	**1**
10	Red LED (2.0-2.8 Vdc)	**1**
11	330 Ohm resistor	**2**
12	TMC2204 stepper driver	**1**
13	Micro-Fit connector 14 pin (female + male)	**2**
14	Micro-Fit connector 4 pin (female + male)	**2**
15	Micro-Fit connector 2 pin (female + male)	**3**
16	Helukabel JZ-500 14x0.5 (or similar flexible 14-lead cable)	**5 m**
17	Dupont/2.54mm connector kit	**1 kit**
18	Stranded wire (20 AWG)	**1 roll**
19	Stranded wire (23 AWG)	**1 roll**
20	Stranded wire (24 AWG)	**1 roll**
21	Power supply (12 Vdc, >= 3 Ampere, 2.5mm DC Barrel Jack connector)	**1**
22	DC Barrel Jack connector (female + male, 2.5mm)	**1**
23	RS232 to TTL Serial Adapter (size should match MR002-001.2)	**1**
24	Kit of Brass Insert Embedment Nuts (see table 2.1 and 4.3)	**1**
25	Kit of screws (see table 2.1 and 4.3)	**1**
26	Double shielded, Ball Bearing 8x22x7	**3**

Table 4.1: *Extruder and mainboard parts to order*

The following are the printed parts for the extruder tool and mainboard housing:

#	Name	QTY
1	Extruder tool chassis	**1**
2	Extruder tool air cooler	**1**
3	Extruder tool servo arm	**1**
4	Extruder tool heat sink holder	**1**
5	Extruder tool cable lock	**1**
6	Extruder computer fan mount	**1**
7	Extruder computer box	**1**
8	Extruder computer lid	**1**
9	Extruder computer mounting plate	**1**
10	Spring covers (hat & shoe)	1

Table 4.2: *Extruder and mainboard printed parts*

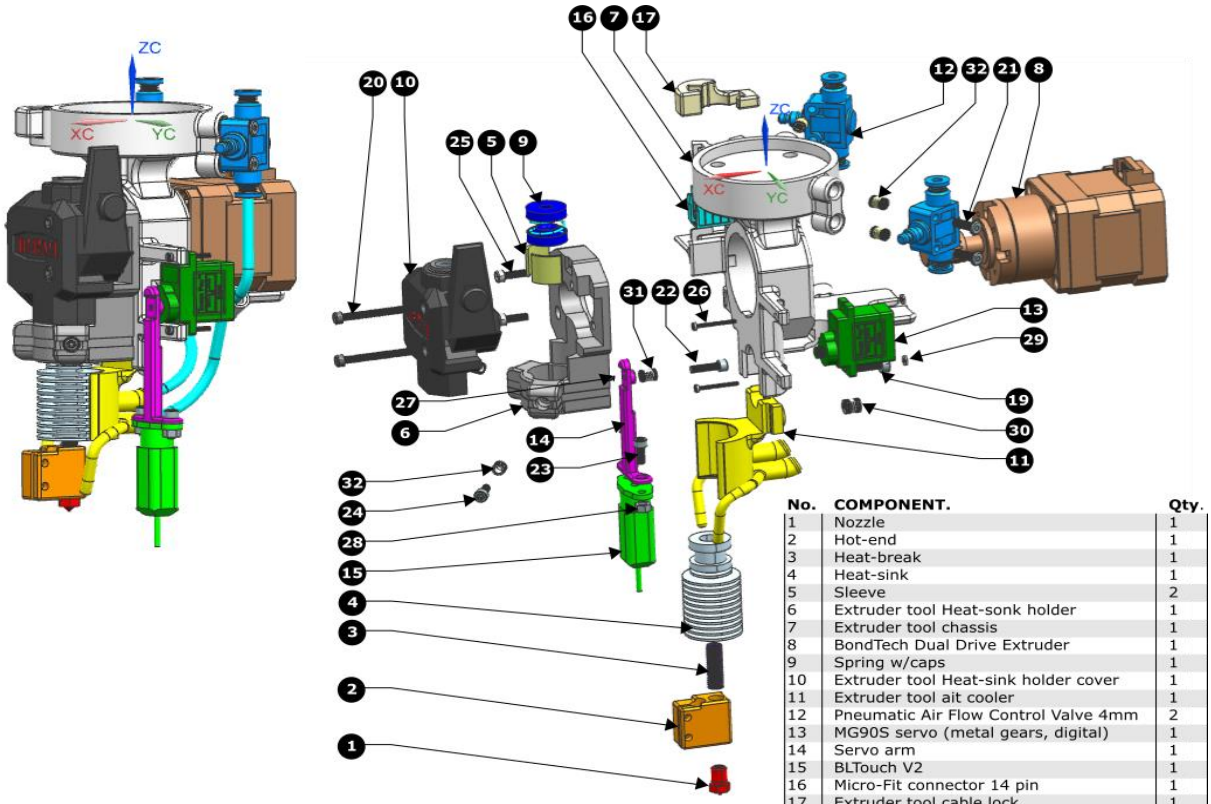
The following parts are required for the spool holder and guide in addition to the printed parts:

No.	Description	Qty.
1	Double shielded, Ball Bearing 8x22x7	**3**
2	M3x9	**3**
3	M6x20	**1**
4	M6 Hex Nut	**1**
5	M3x7 washer	**3**
6	M3x4x5 Brass Insert Embedment Nut	**3**

Table 4.3: *Spool holder and filament guide parts to order.*

4.2 Extruder tool assembly guide

Figure 4.1 shows the full assembly sequence of the extruder tool end-effector.

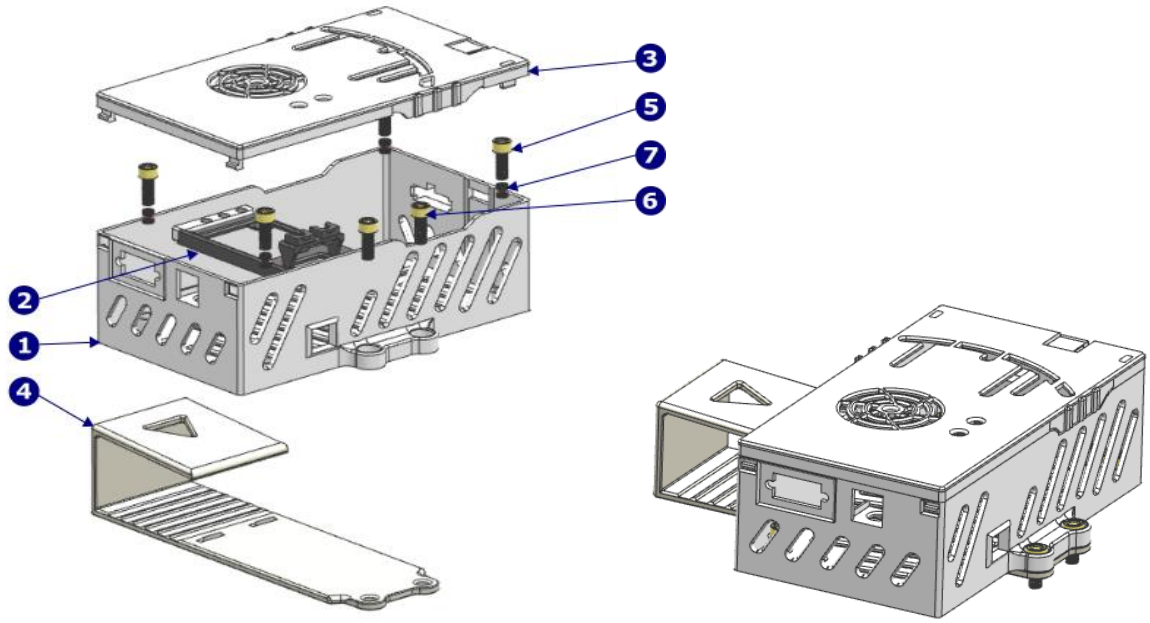


No.	COMPONENT.	Qty.
1	Nozzle	1
2	Hot-end	1
3	Heat-break	1
4	Heat-sink	1
5	Sleeve	2
6	Extruder tool Heat-sonk holder	1
7	Extruder tool chassis	1
8	BondTech Dual Drive Extruder	1
9	Spring w/caps	1
10	Extruder tool Heat-sink holder cover	1
11	Extruder tool ait cooler	1
12	Pneumatic Air Flow Control Valve 4mm	2
13	MG90S servo (metal gears, digital)	1
14	Servo arm	1
15	BLTouch V2	1
16	Micro-Fit connector 14 pin	1
17	Extruder tool cable lock	1
18	M5x12 (note)	4
19	M4x10	1
20	M3x29	2
21	M3x18	4
22	M3x12	1
23	M3x9	2
24	M3x8	2
25	M3x7	2
26	M2x16	2
27	M1.5x3	1
28	M3 Hex nut	2
29	M2 Hex nut	2
30	M4x4x6 Brass Insert Embedment Insert	1
31	M3x6x5 Brass Insert Embedment Insert	1
32	M3x4x5 Brass Insert Embedment Insert	6

Figure 4.1: Extruder tool assembly sequence

4.3 Mainboard housing assembly guide

The housing for the extruder mainboard is made up of four main printed components. It is simple to put together and can be altered based on the connection configuration. See figure 4.2.



No.	Component	Qty.
1	Extruder computer box	1
2	Extruder computer fan mount	1
3	Extruder computer lid	1
4	Extruder computer mounting plate	1
5	M4x10	4
6	M5x14	2
7	M4x6x6 Brass Insert Embedment Nut	4

Figure 4.2: *Mainboard housing assembly*

The MKS Gen 1.4 mainboard is connected to the extruder tool through a flexible 14 lead cable. The cable is connected to a 14-pin Micro-Fit connector on both the mainboard and extruder tool sides. All the wiring between the 14-pin connector and the mainboard uses smaller Micro-Fit connectors to allow for easy part replacement. The same applies on the extruder tool side.

Figure 4.3 show the input and output pins on the MKS Gen 1.4 mainboard.

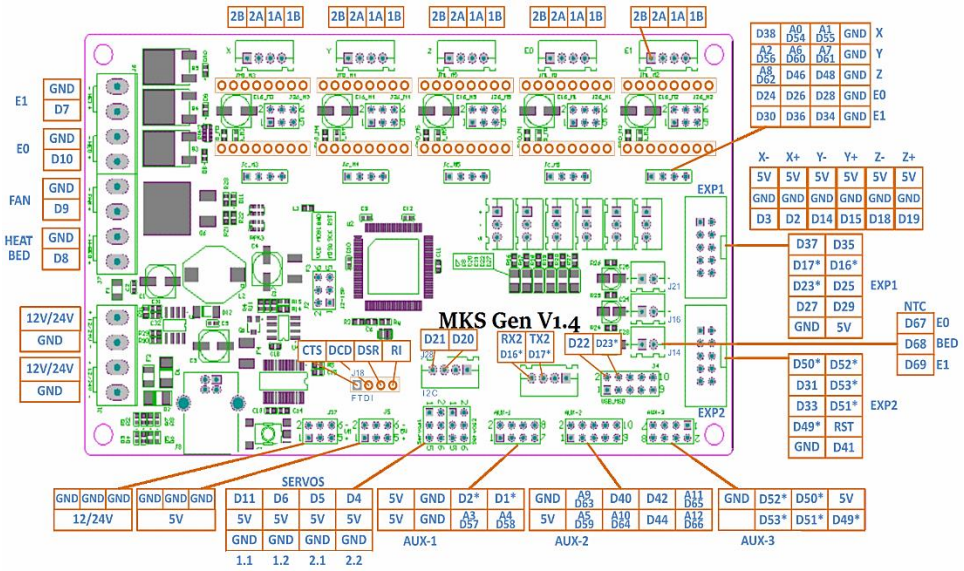


Figure 4.3: Mainboard pinout

Table 4.3 show how to connect the mainboard pins to the extruder tool electrical components. Care must be taken to connect the correct wire on both sides of the 14-pin connectors going between the mainboard housing and extruder tool.

Component		Signal	Pin	Old pin	Notes
Hotend		V+	D10 (E0+)	D3	Polarity N/A
		V-	GND (E0-)	GND	
Thermistor		T	D67 (A13)	A0	Polarity N/A
		GND	GND	GND	
BL Touch	Servo	S	D6	-	
		5V	5V	-	
		GND	GND	-	
	Prove	Z-	D18 (S)	-	
		GND	GND (-)	-	
LEDs	Green	S	D5	-	
		GND	GND	-	
	Red	S	D4	-	
		GND	GND	-	
Mount servo		S	D11	-	
		5V	5V	-	
		GND	GND	-	
Stepper		1B	1B	1B	Some motors may have different pin arrangements. Switching the two middle cables usually solves the problems.
		1A	1A	1A	
		2A	2A	2A	
		2B	2B	2B	

Table 4.3: Mainboard to extruder tool connections

Chapter 5

RAM Quick Start

The following guide will in detail explain how to prepare and run the robotic additive manufacturing platform created in the project.

5.1 Robot preparations

Before working with the robot, always ensure that the compressed air supply is enabled to the robot.

The initial step in preparing the RAM platform is changing the robot end-effector. The end-effector attaches to the tool flange connected to the last joint of the robot. The flange is controlled pneumatically using a digital output signal on the robot controller. This signal may change in name.

If there is another end-effector connected to the robot, it should be carefully disconnected and removed by experienced personnel.

Once the extruder tool has been attached to the flange, connect the flexible cable between the extruder tool and mainboard housing through the plastic cable. Ensure that the cable has enough give to allow the end-effector to rotate relatively freely, but not so much that it gets in the way of the hotend.

Ensure that the extruder tool heatsink valve is not closed, as this could cause the hotend to melt.

Connect the serial cable going from the robot controller to the mainboard housing and connect the mainboard box power supply to a wall outlet.

Select which build platform to use and put it on the table in front of the robot. The platform must be suitable for additive manufacturing and should be as flat as possible.

5.2 Tooldata calibration

If the extruder tool has been changed in any way lately which changes its tooltip positions, such as replacing the hotend, the tooldata variables of both the touch probe and the hotend (`tProbe` and `tExtruder` respectively) MUST be re-defined to reflect those changes. Failing to do so may cause the tooltips to crash.

Please check the robot user manual how to define end-effector tooltips. Define `tProbe` to match the touch probe tip and `tExtruder` to match the hotend tip (see figure 5.1).

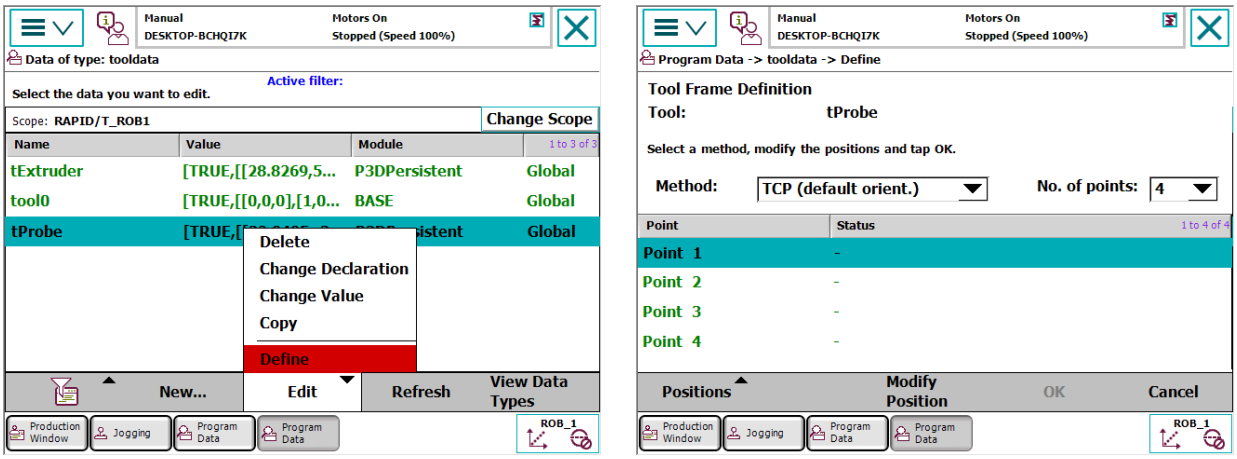


Figure 5.1: *FlexPendant tooldata definition*

After the tooldata variables have been redefined, the `nProbeOffsetMm` variable must be updated to reflect the new height offset between the two tooldata definitions `tProbe` and `tExtruder`. See chapter 5.3 for more information.

5.3 RAPID configuration

Open RobotStudio on the local computer and add the network robot controller to the list of controllers. In the RAPID code tab, all the robot modules will be listed. Ensure that the `P3D*.sys` modules are available.

Ensure that the only command inside the `main` procedure is "`StartP3D;`". This is the main entry point of the AM robot program.

All configurable parameters in the robot program are available in the `P3DProfile.sys` module. The default values are fine for most of the values. Some of the parameters which may probably need to be updated are:

- `nPrinterSizeX/Y/ZMm`: These define the size of the available printing area. It should be smaller than the total size of the build platform to allow for a decent margin between the edge of the platform and the print coordinate system origin point. `Z` is the vertical component and should be at a reasonable height.
- `nProbeOffsetMm`: If the tooldata has been updated, this value needs to be updated to reflect the height difference between the probe and hotend tooldata definitions and their actual tooltips. This is a tricky value to get right. If the error is small, it can be adjusted by trial and error by manually measuring the height of the first printed layer compared with the expected first layer height and adjusting `nProbeOffsetMm` by the error.

5.4 Print bed definition and leveling

The print bed coordinate system `wBed` must be defined to reflect the new print surface. This is done

by using a 3-point definition of the `wBed` variable of type `wobjdata`. The three points are three of the build platforms corners. The corners are as follows:

- `X0`: The origin of the print coordinate system.
- `X1`: Seen from above, the corner **counterclockwise** from `X0`.
- `Y`: Seen from above, the corner **clockwise** from `X0`.

The touch probe is used as the reference point when defining the build platform corners. Open the jogging menu on the FlexPendant and use the settings in figure 5.2.

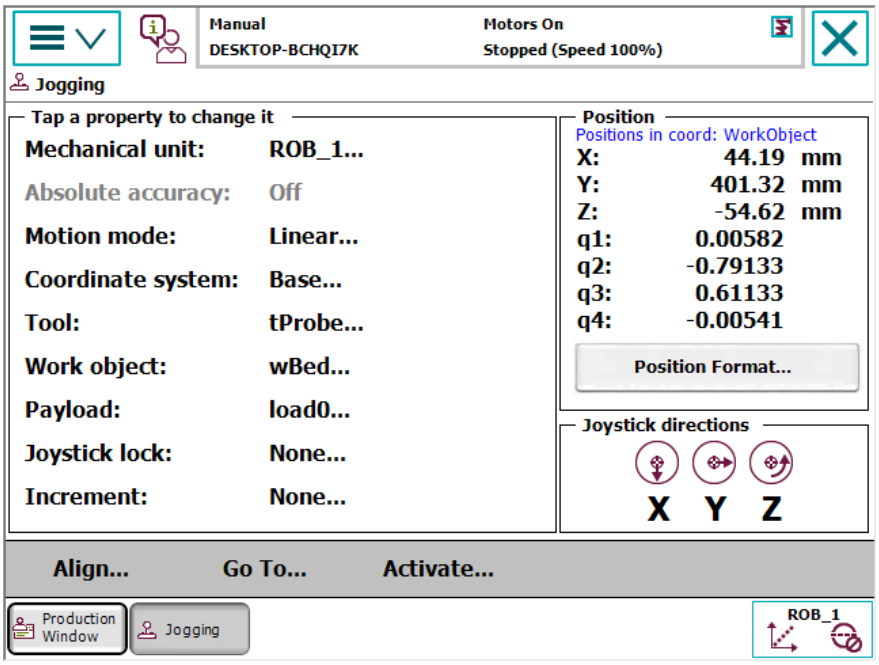


Figure 5.2: *FlexPendant jogging screen*

Make sure the tool is aligned vertically to the world coordinate system. To deploy the probe, run the robot program in manual mode and select `Extruder menu -> Descend arm with probe` like in figure 5.3. Quickly stop the program after the probe is deployed line in figure 5.4.

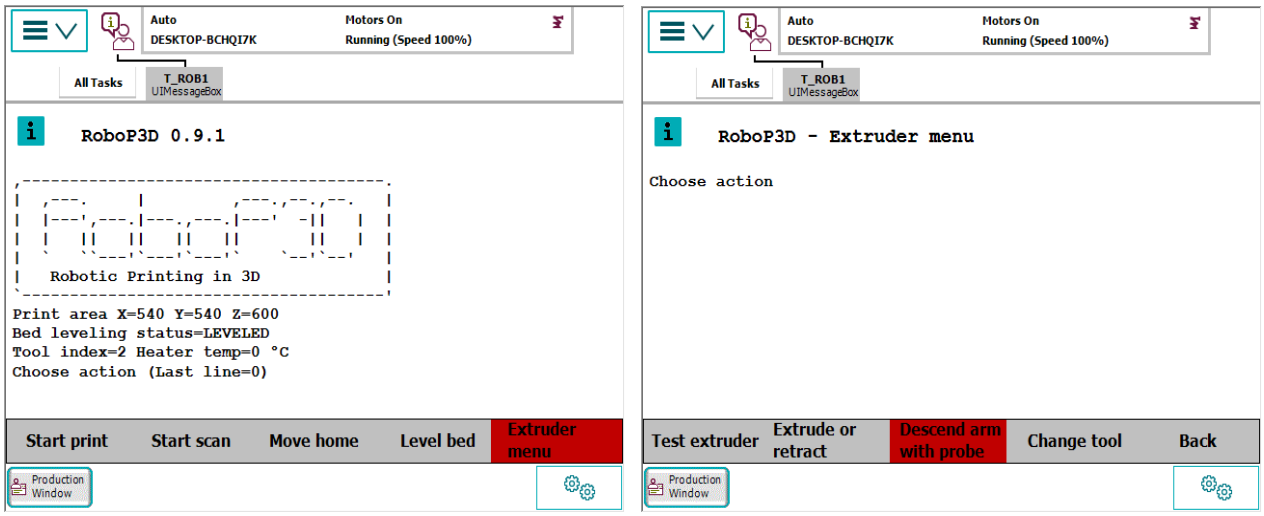


Figure 5.3: *FlexPendant probe descend*

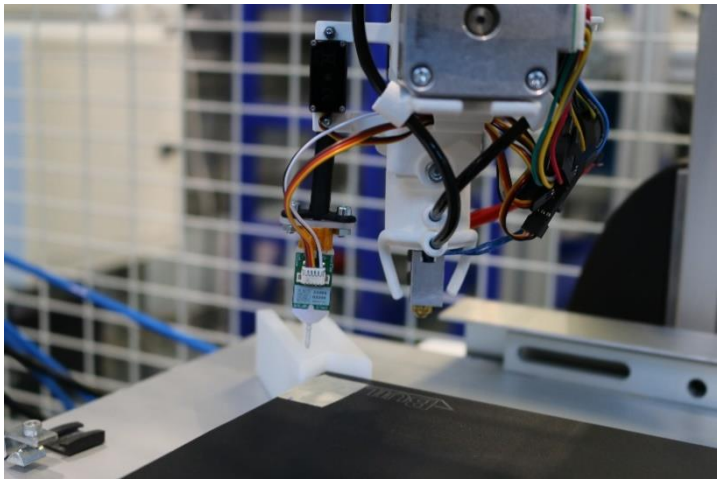


Figure 5.4: *Deployed touch probe*

Open the `wobjdata` menu on the FlexPendant and start defining `wBed` using "`3 points`" like in figure 5.5.

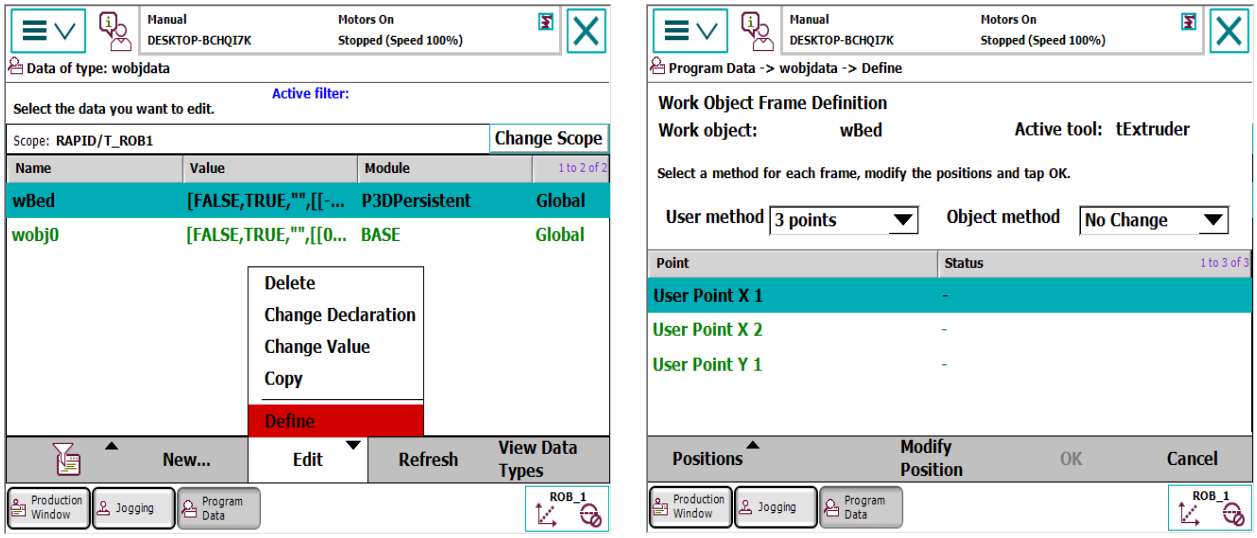


Figure 5.5: *FlexPendant wBed definition*

Navigate the probe tip just inside and above each corner of the build platform, then run `Extruder menu -> Descend arm with probe` again. This time let the robot finish the probing procedure. When the final position has been reached, modify the position of the correct `User Point` in the Work Object Frame Definition menu. Do this for each of the three corners and then click `Ok` to finish the bed definition.

Once the build platform is defined, run the robot program and select `Level bed` like in figure 5.6. This command will make the robot create a `NxN` bed leveling grid across the build platform. A message will show up on the FlexPendant if the bed leveling was successful.

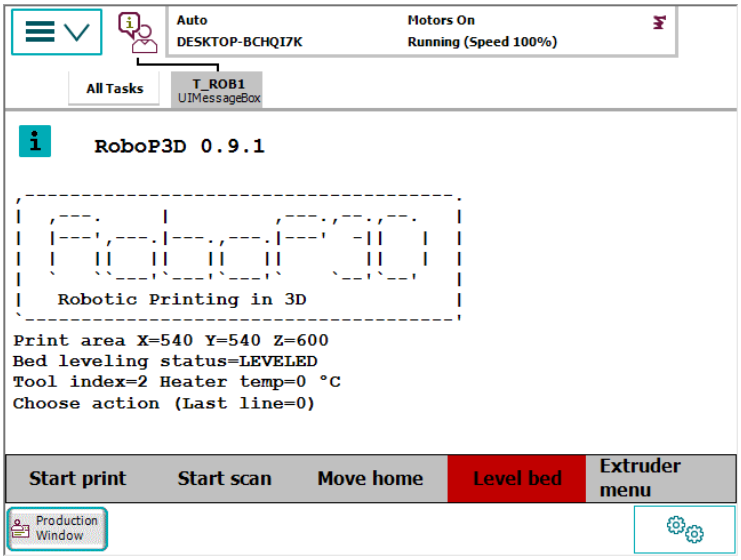


Figure 5.6: *FlexPendant run bed leveling*

5.6 Changing filament

To load a new filament into the extruder, run the robot program in manual mode and select `Extruder > Extrude or Retract` like in figure 5.8. This allows the user to unload the old filament and load new filament into the extruder.

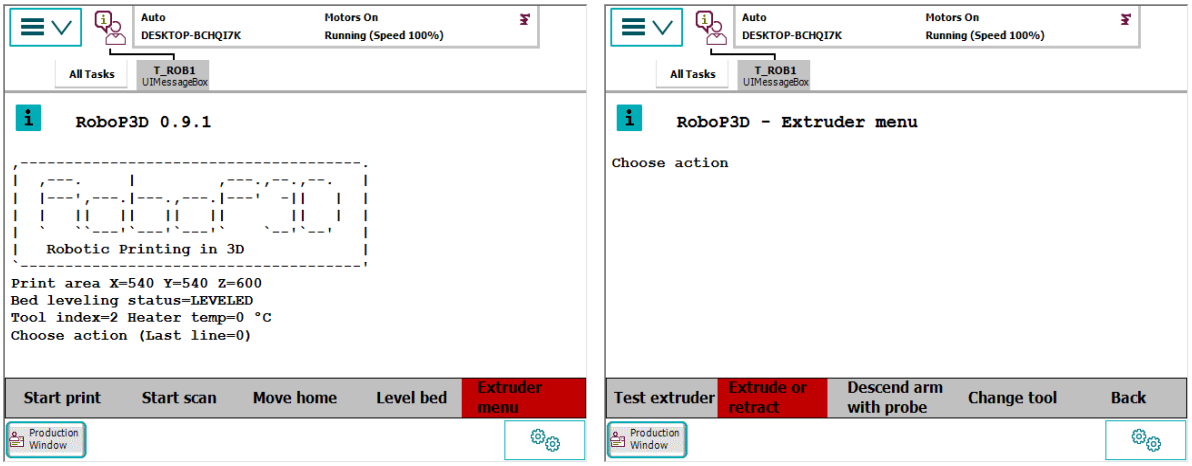


Figure 5.8: *FlexPendant extrude*

5.7 Starting a new print

To create a G-Code file from a model to print, open PrusaSlicer and select the correct RAM profiles based on the current material type, nozzle size and filament diameter. See chapter 6.3 for more details.

Once the G-code file has been created, it can be transferred to the robot controller using RobotStudio using the `File Transfer` button in the `Controller` tab.

Once the file is on the controller, run the robot program again in automatic mode, and select `Start Print` like in figure 5.7.

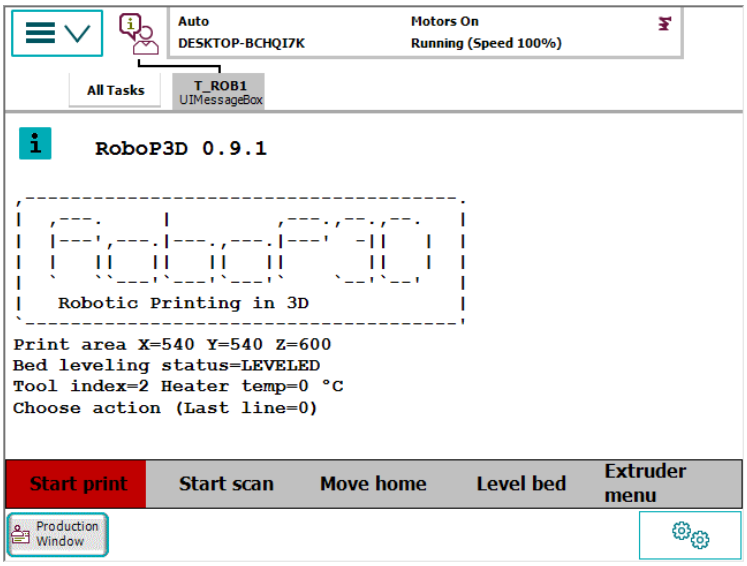


Figure 5.7: *FlexPendant start print*

WARNINGS

Follow the print very carefully at the start of the print to ensure everything is in order. Stop the print immediately if:

- it looks like the hotend is scraping the platform.
- it looks like the hotend is bouncing and wiggling against the printed object.
- in freeform printing it looks like any of the cables are pulled too tight or if the arm is approaching a singularity point which may cause a rapid turning motion. See chapter 3.3.2.
- if the printed object warps and lifts from the platform
- if the printed object detaches from the platform
- anything else looks out of the ordinary

Chapter 6

Materials, slicers and test prints

This chapter will describe the different printing materials and slicing software used within the project, as well as showcase a number test and calibration prints.

6.1 Printing materials

The materials used in additive manufacturing come in various formats. Thermoplastic feedstock often come in the shape of 3D printer filament. The filament is made from a continuous plastic thread spool (see figure 6.1), which is pulled/push through an extruder motor and extruded at the heated hot end.

Figure 6.1: *Spool of PLA 3D printer filament*

Several printing materials have been evaluated and tested with the robotic additive manufacturing platform in the project. Included in these materials are:

- PLA (polylactic acid): One of the most common 3D printing materials. PLA melts at a relatively low temperature. The material is accessible and easy to work with.
- ABS (Acrylonitrile butadiene styrene): Another common 3D printing material. ABS melts at a higher temperature than PLA. Due to warping, ABS prints generally require a heated bed for a successful print. ABS is stronger than PLA.
- PETG (Polyethylene terephthalate glycol): A material which has grown in popularity in the last years. It has a higher melting temperature than PLA and is such more stable at a higher temperature. PETG is also relatively stable with a low shrink ratio.
- PETT (Polyethylene coTrimethylene Terephthalate): Is a special type of PET which is semi-transparent.
- PP (Polypropylene): PP is a relatively soft in comparison to the other materials. PP is resistant to some chemicals. Packing tape was used as a print surface to successfully make the PP adhere to the build platform.

A few additional materials were attempted but had skipped due to various reasons. These include:

- Recreus Filaflex: A flexible material used to make flexible prints. Due to the way the filament bends, it does not allow for any openings inside the extruder as the filament will easily bend inside extruder rather than being fed out the hot end.
- Wooden PLA: This is a special type of PLA material with integrated wooden fibers. This material was attempted but cancelled due to the amount of smoke caused by the large hot end.

Multiple calibration prints were created using the different materials. The slicer profiles had to be updated based on various parameters for each material.

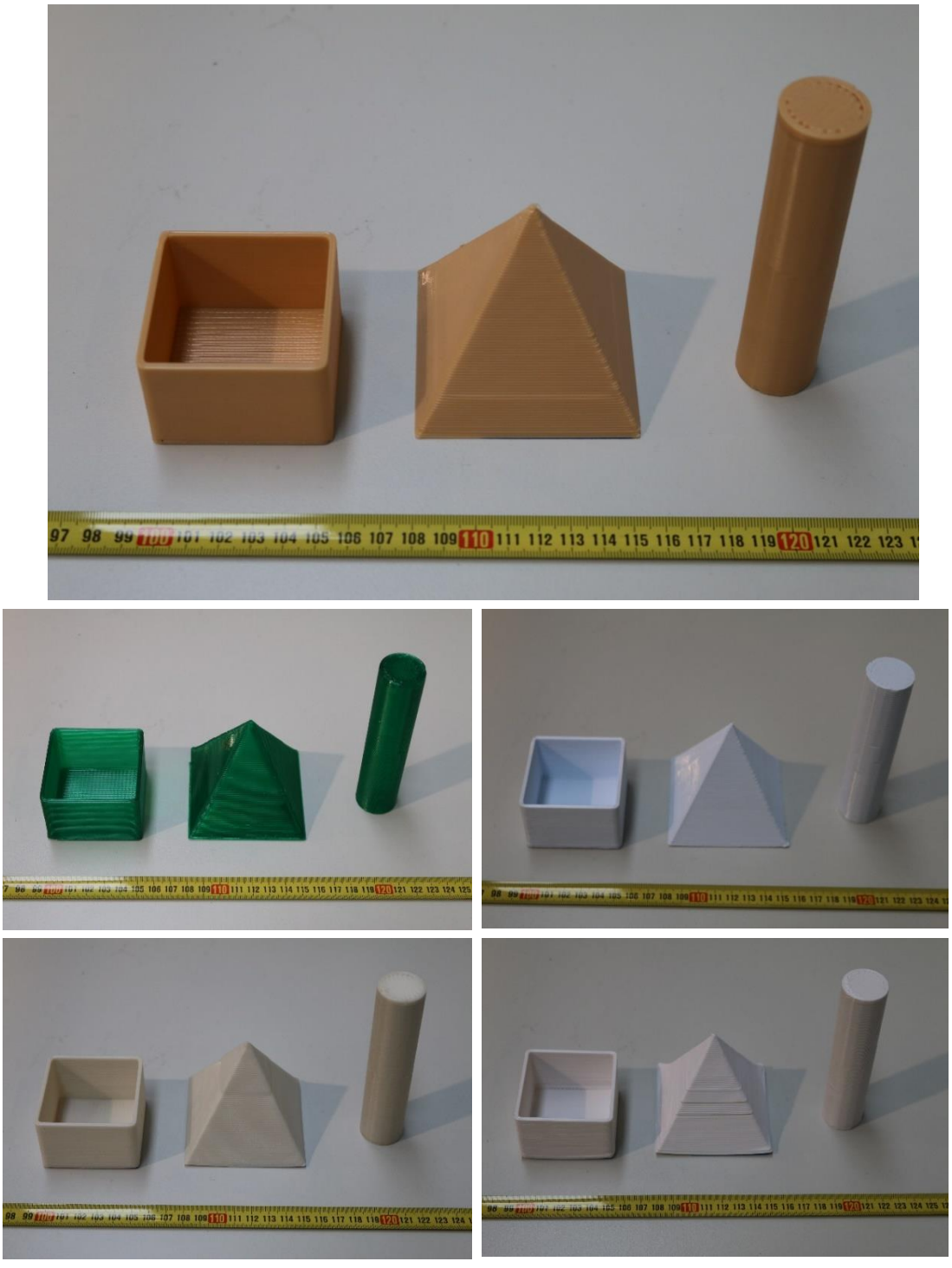


Figure 6.2: *Calibration prints in multiple materials*

6.2 Slicing software

Other than the custom slicing software used for freeform printing, several widely available open-source slicing programs were used to create G-Code for RAM. These were Ultimaker Cura and PrusaSlicer (based on Slic3r).

Cura was used at the start of the project as it is one of the most popular and fully featured slicing programs available. The prints created using Cura were successful based on the slicing settings.

Over the course of the project, we switched to using PrusaSlicer more than Cura. While Cura has more features and quality of life improvements over PrusaSlicer, the simplicity of PrusaSlicer seemingly generated less complex G-Code which contained a smaller number of short quick movements. This helped significantly with the issues described in chapter 3.3.5. The result was less critical errors and failed prints.

It must be noted that this is a unique scenario, which does not apply to typical 3D printers. The fine-detail quality of the printed objects was of less importance than the size of the object and the reliability of the print process.

6.3 Calibration prints

During the project, many calibration prints have been created for various calibration purposes. These prints were created to:

- Find errors and optimizations in the robot program.
- Adjust the print speed based on nozzle size (see figure 6.3).
- Adjust the part cooling speed.
- Optimizing slicer settings for printing using a robotic arm.

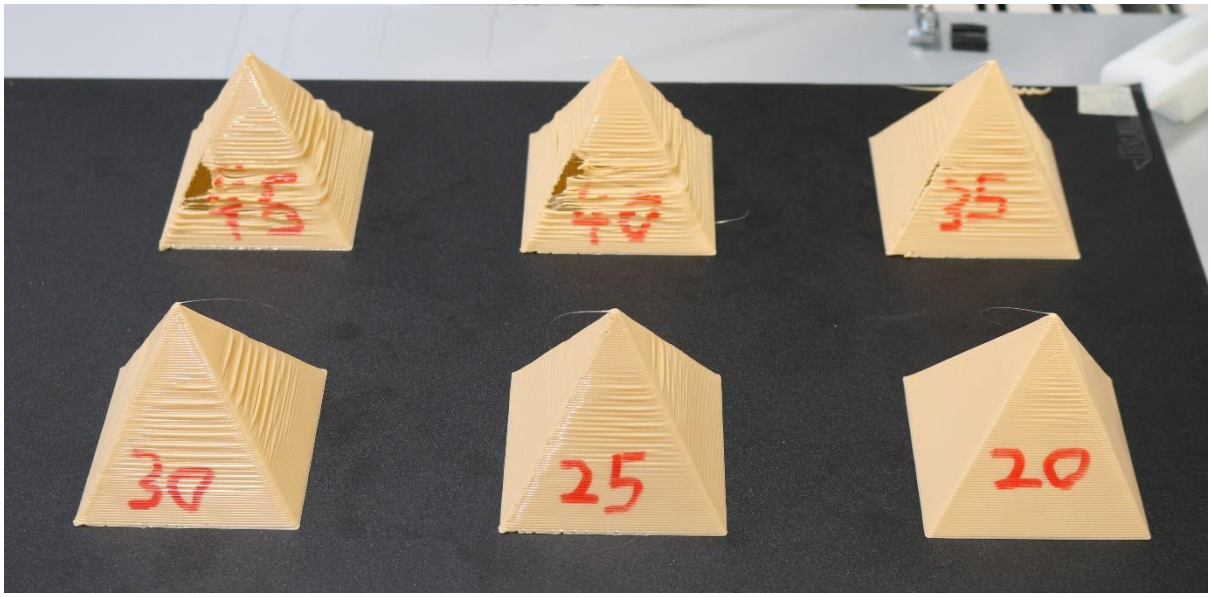


Figure 6.3: *Print speed comparison in millimeters per second*

6.3 PrusaSlicer profiles

PrusaSlicer profiles defines the different settings used for different types of materials, nozzle sizes, 3D printers and for print quality in general. Many different optimizations and profiles were tested in the RAM project for both 1.75 mm and 2.85 mm filament diameter.

There are some notable differences between these profiles and profiles used with typical 3D printers. These include but are not limited to:

- Always lift Z during retraction move to minimize scraping the printed object.
- The minimum detail resolution has been significantly increased to reduce the number of small moves.
- PrusaSlicer was set to complete individual objects before moving on to the next object. The robotic arm has a lot of room to print complete objects without worrying about collisions.

Another feature of PrusaSlicer which was extensively tested was the so-called Vase mode (see figure 6.3). In this mode a single-line wall is printed for the entire object in a long vertical spiral. This mode has the benefit of being ease and reliable to print due to the simplicity of the G-Code. The mode is only viable for certain types of models, such as vases and other circular or polygonal objects without the need for flat top layers.

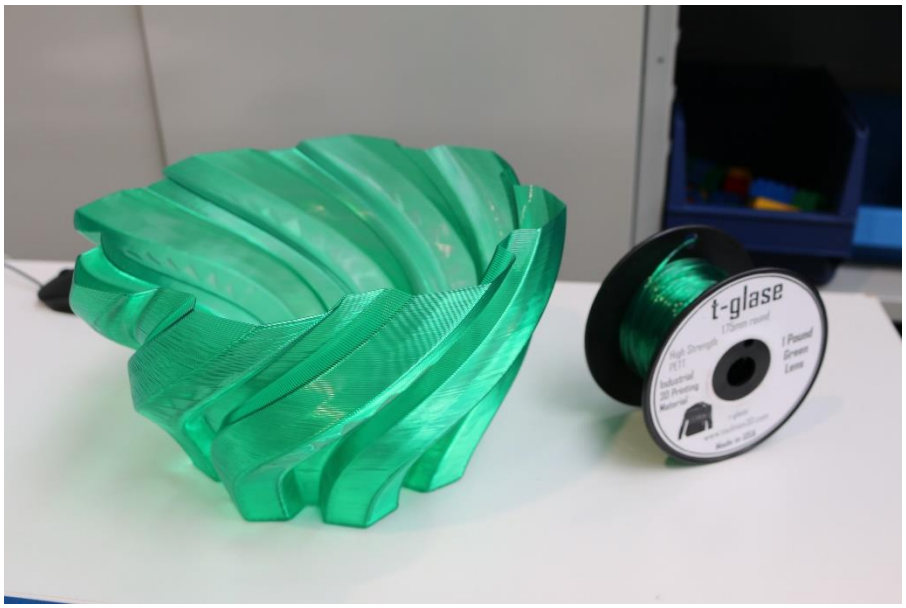


Figure 6.3: *Vase-mode in PETT*

6.4 Print showcase

Over the course of the project there have been many successfully printed objects using the robotic additive manufacturing platform, both small and precise objects and very large objects. Figure 6.4 shows a small sample of these.

Figure 6.4: *Print showcase*