1 Design

1.1 Mechanics

The mechanical design of the extrusion force test bench can be divided into the following parts:

- Frame of aluminum profiles
- Milled aluminum mounts for the extruder and the Hotend
- Filament extruder
- Hotend with fan
- Load Cell
- Filament length encoder

1.1.1 Individual parts

Stepper Motor

To push the filament through the hotend the E3D Rapidchange Revo Hemera-1,75mm Extruder was used. The stepper motor works with 12-24V and can (depending on the filament) generate an extrusion force up to 120N.

Figure 1: Extruder

Mosquito Magnum Hotend

The Mosquito Magnum hotend was chosen because it can reach Temperatures up to 450°C, it can be driven with high printing speed, and its nozzles can be changed easily.

Figure 2: Mosquito Magnum Hotend

Load Cell

In order to achieve controlled deformation,a load cell capable of carraying a maximum mass of 20kg was used

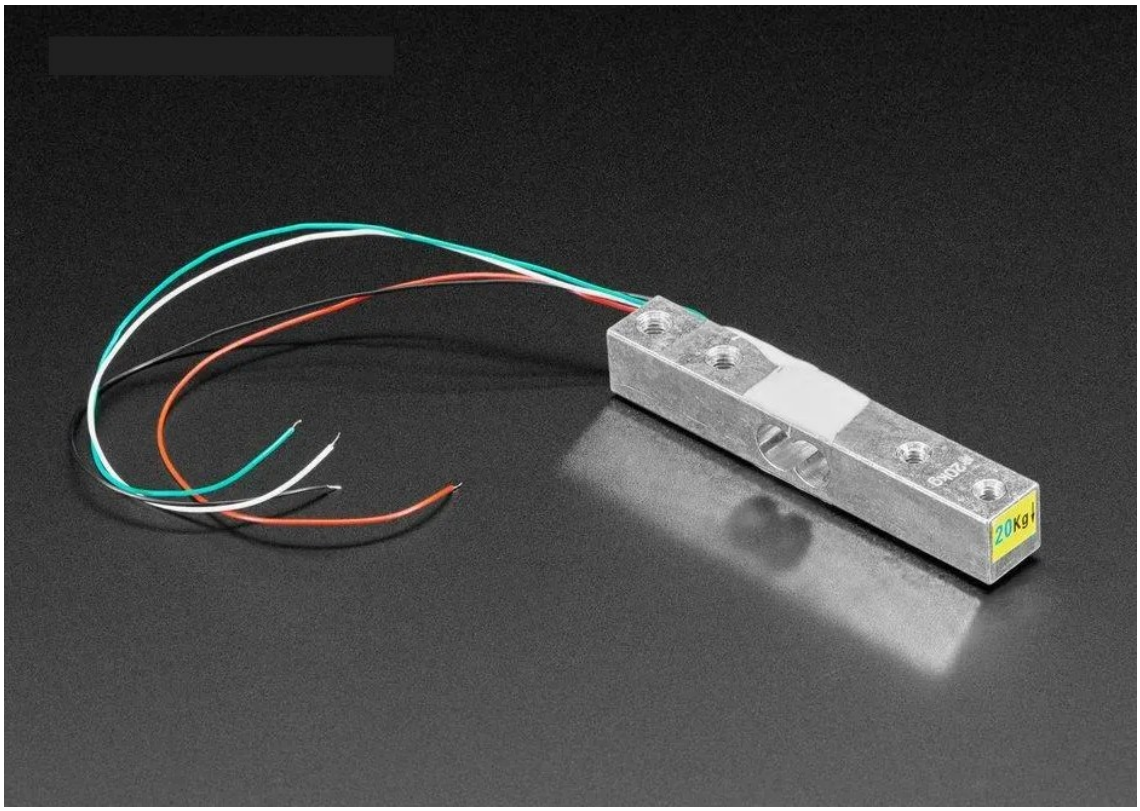


Figure 3: Load Cell

Filament Length Encoder

To measure the length of filament which is fed in the hotend the Orthus Filament Monitor was used.

Figure 4: Filament Length Sensor

Filament Length Encoder Mount

To hold the Encoder in its place a 3D printable mount was constructed.

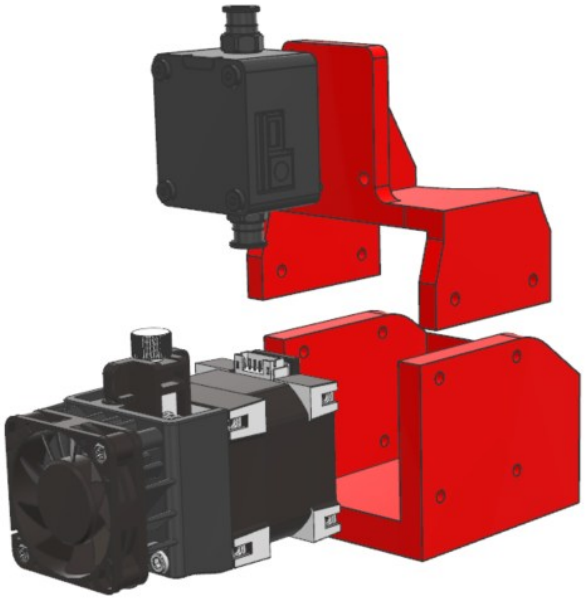


Figure 5: Filament Length Encoder

1.1.2 The first prototype

The most important goal of the whole design was enabling a deformation of the load cell caused by the extrusion force between extruder and hotend.

To be able to measure the extrusion force of the extruder, it was necessary to split the extruder and the hotend. To prevent the filament from buckling, the design of the load cell, extruder and hotend mounts were made so that the distance between the extruder and hotend is as small as possible.

With this design the load cell can be bent downwards by the extrusion force which arises between the hotend and the extruder.

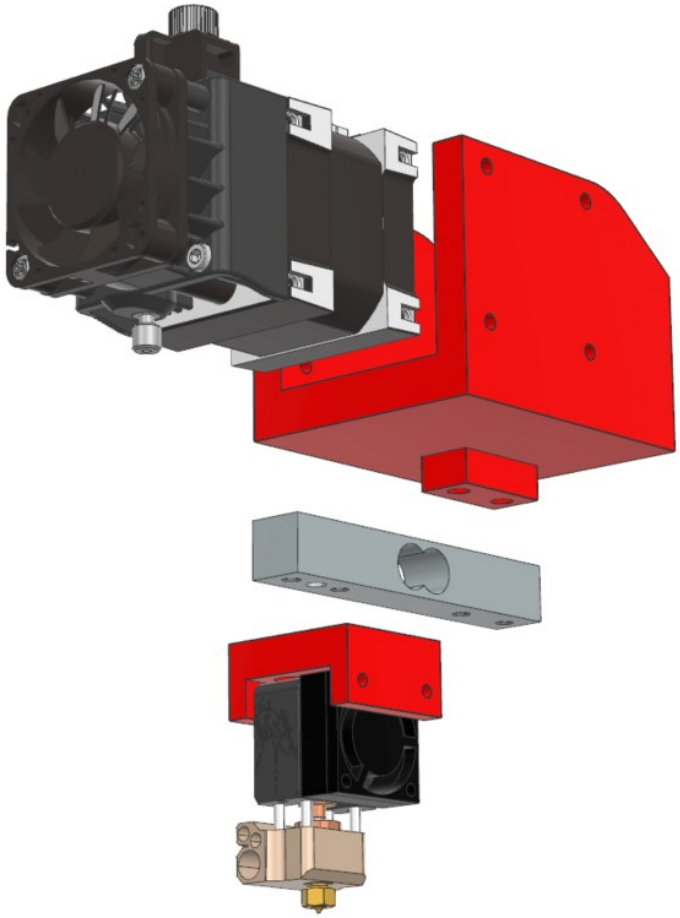


Figure 6: Modell of the first prototype

The first prototype of this construction was printed with PLA, to see if every part fits in its place and the whole idea of deforming the load cell works in such a way.

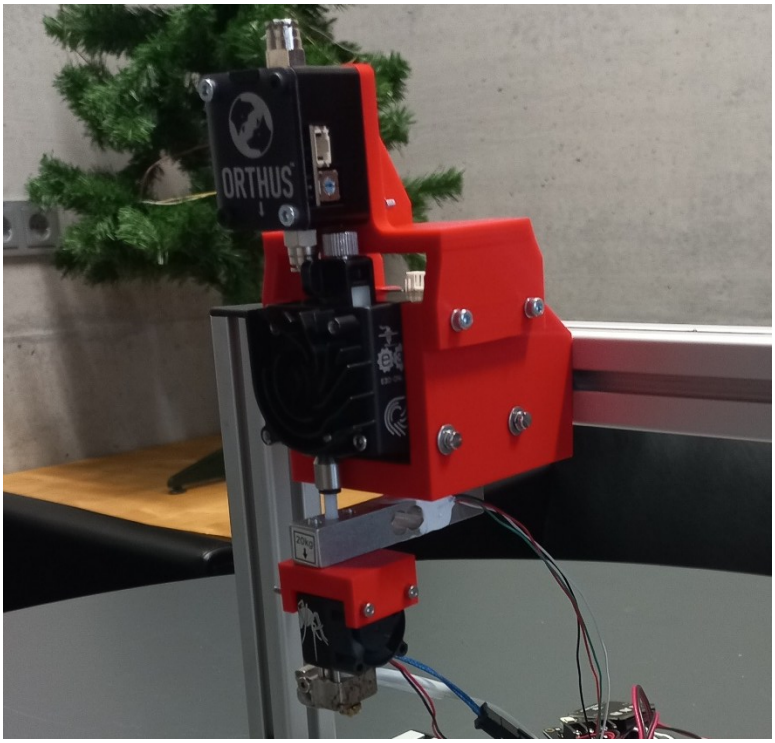


Figure 7: Prototype of PLA

After the first prototype had been proven to work, the mount for the load cell and extruder was redesigned so that it could be milled from aluminum to achieve better overall stiffness properties and eliminate the temperature behavior that affects the stiffness properties of PLA. The holder for the hotend should also be milled from aluminum.

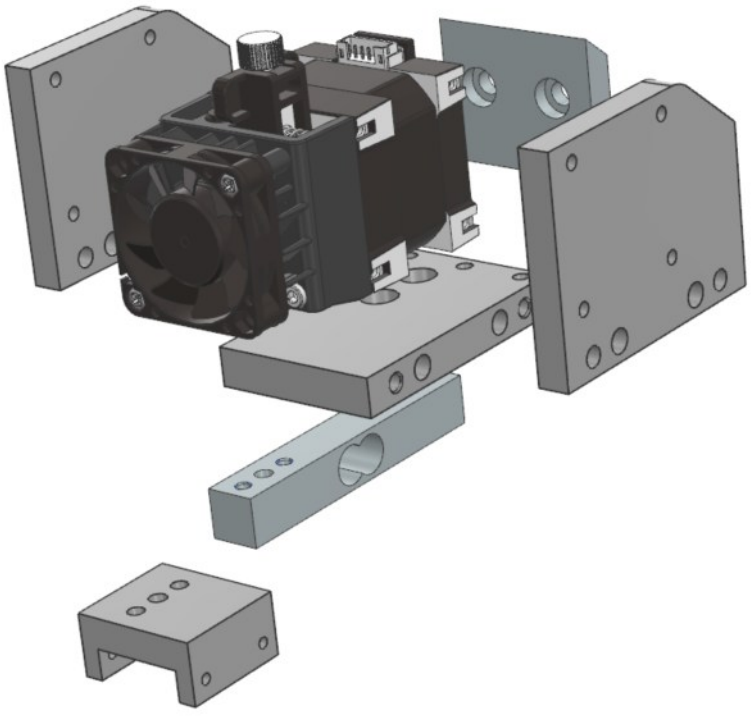


Figure 8: Adaptation design for milled part

1.1.3 The Final Version

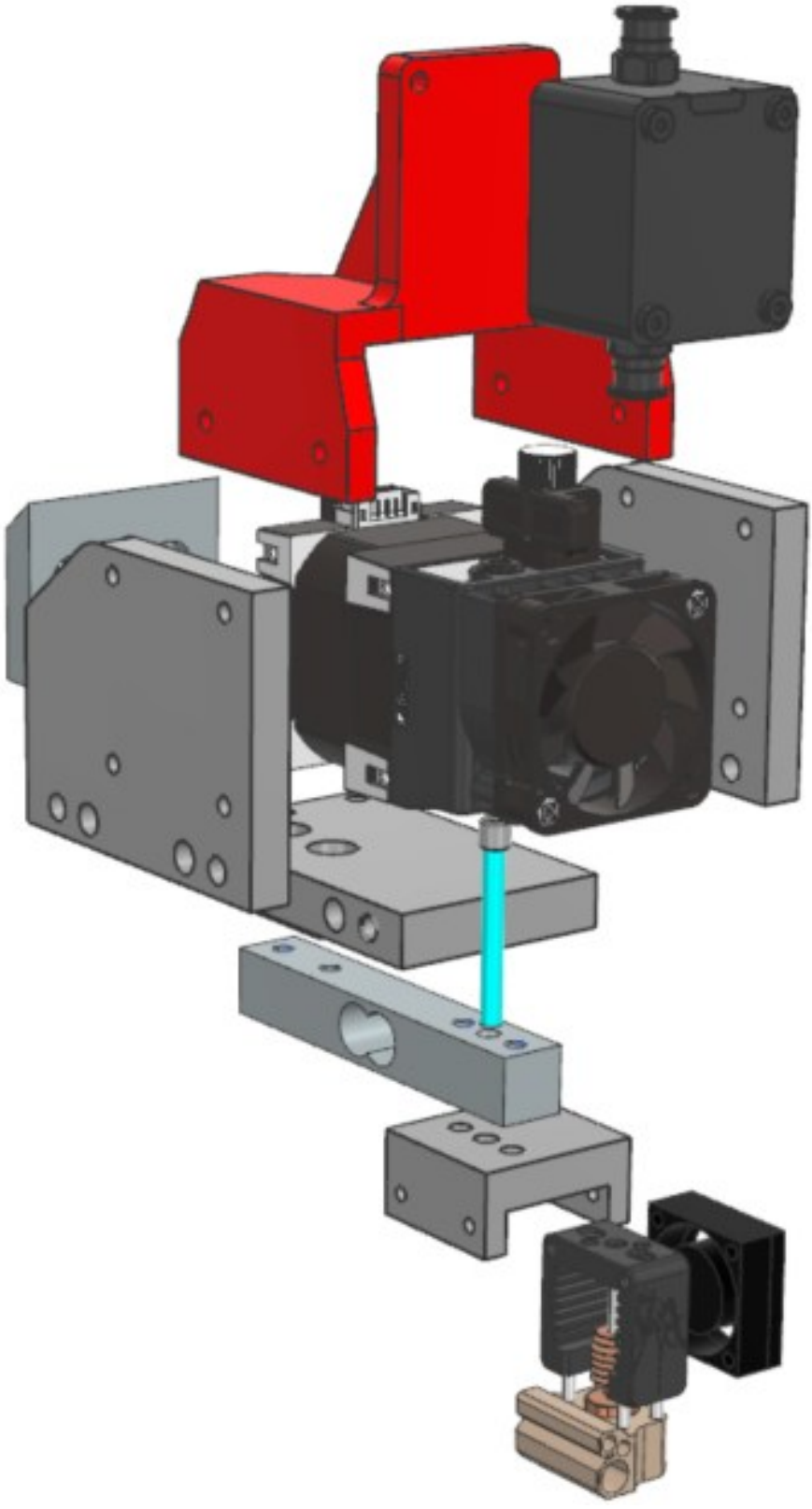


Figure 9: Final Modell exploded View

In this picture, you can see all the parts of the final version except for the aluminum profile frame.

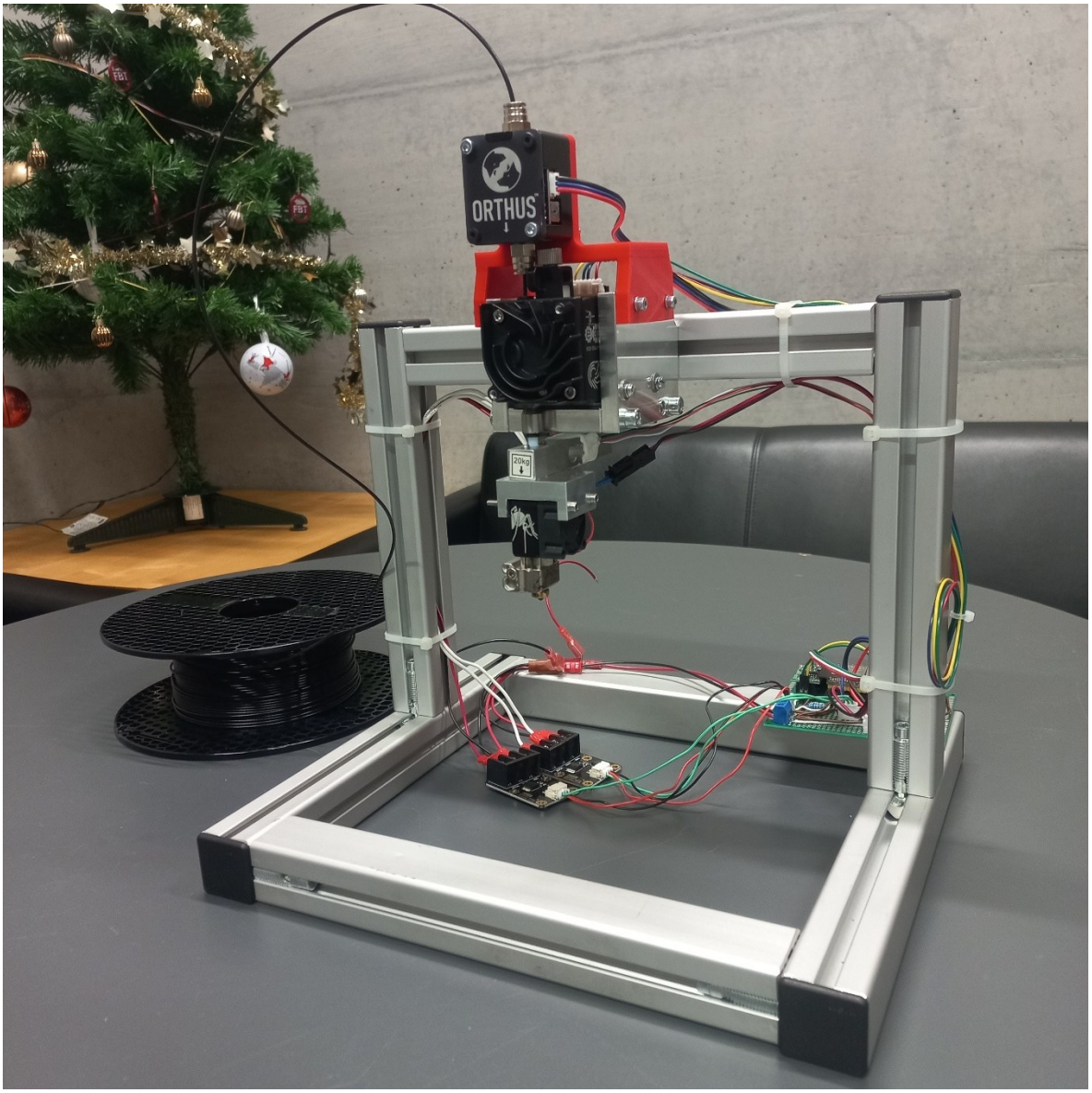


Figure 10: Final version of the test bench

Here we can see the final assembly of the test bench with the extruder, filament length encoder, load cell, and the hotend mounted on the aluminum frame.

1.2 Electronics

1.2.1 Individual Components

- ESP32-S3-DevKitC-1
- HX711 Amplifier with ADC
- Prusa Heating Resistor (24V, 40W)
- NTC-Thermistor
- TMC2209 V1.3 Stepper Motor Driver
- Filament length encoder
- Gravity MOSFET Power Controller

To run the embedded software, an ESP32-S3-DevKitC-1 was chosen because of its many pins and its two Cores to run real multi-threading.

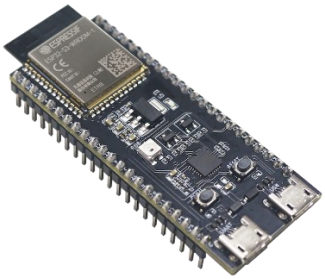


Figure 11: Microcontroller

To measure the force with which the load cell is deformed, a HX711 24-bit analog digital converter was used. It is necessary to use this special IC because the change in voltage is caused by a very small change of resistance of the strain gauges on the load cell. The HX711 amplifies this small voltage with a wheatstone bridge and operational amplifier and converts it into a digital signal with a resolution of 24 bits.

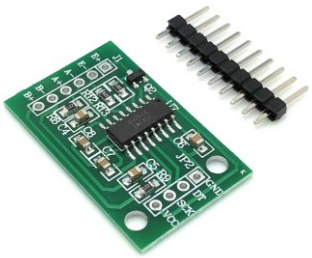


Figure 12: HX711 Amplifier

To heat up the hotend a Prusa heating resistor with 24V and 40 Watt was used. This heating resistor is controlled by a Gravity MOSFET Power Controller, which is a module that contains a MOSFET, a gate driver which can be operated with the logic level of an MCU (3,3V) and a freewheeling diode. The fan which cools the hotend is also controlled by such a MOSFET module.

Figure 13: Prusa Heating Resistor

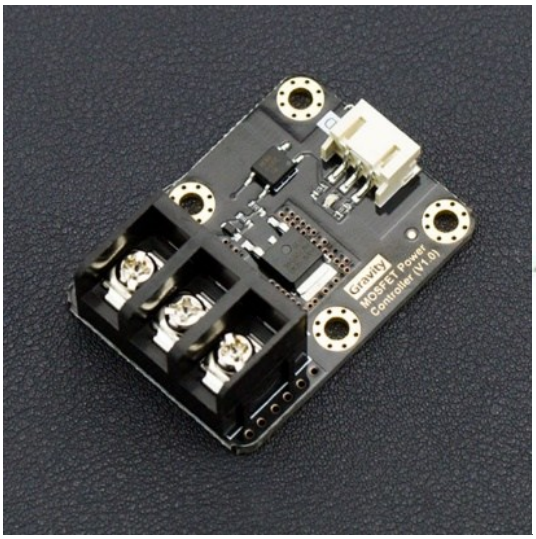
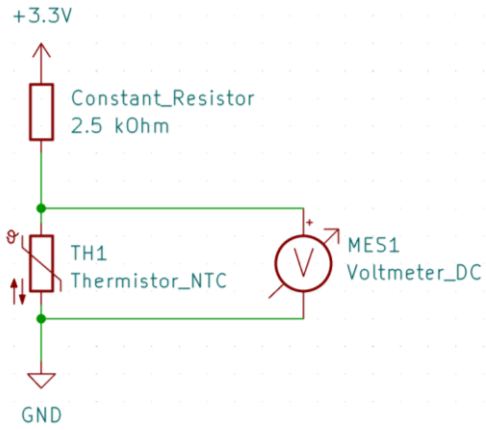


Figure 14: Gravity MOSFET Power Controller

The temperature at the hotend is measured by an NTC Thermistor. To relate the temperature at the hotend to a voltage, which is measured by an intern ADC of the ESP32, the decision was made to use a voltage divider. This voltage divider was designed with 2,5 kOhm resistor and the NTC Thermistor in series. The voltmeter in the following picture should represent the ADC which measures the voltage at the resistor to ground.

Figure 15: Thermistor Circuit



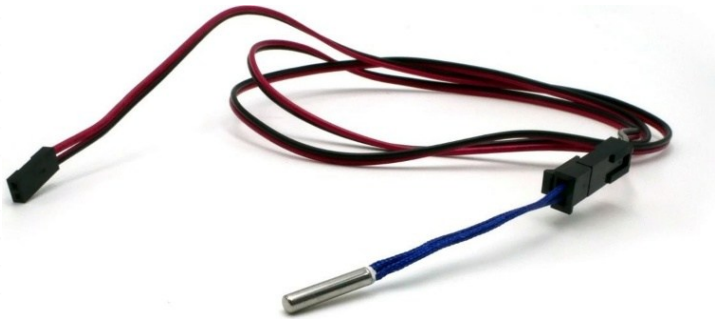


Figure 16: Thermistor

The stepper motor of the extruder needs to be controlled by a stepper motor driver. To accomplish this a TMC2209 V1.3 stepper driver was used because of its micro stepping ability and the smooth way it can run the stepper motor.

The driver is operated in a micro stepping setting of 8 micro steps. Which means that a full step (which would be done normally) can be done in 8 micro steps. The stepper motor has 200 full steps for whole rotation. In our micro stepping setup, a whole rotation can be done in 1600 steps which leads to a fine resolution and a smooth silent run of the stepper motor.

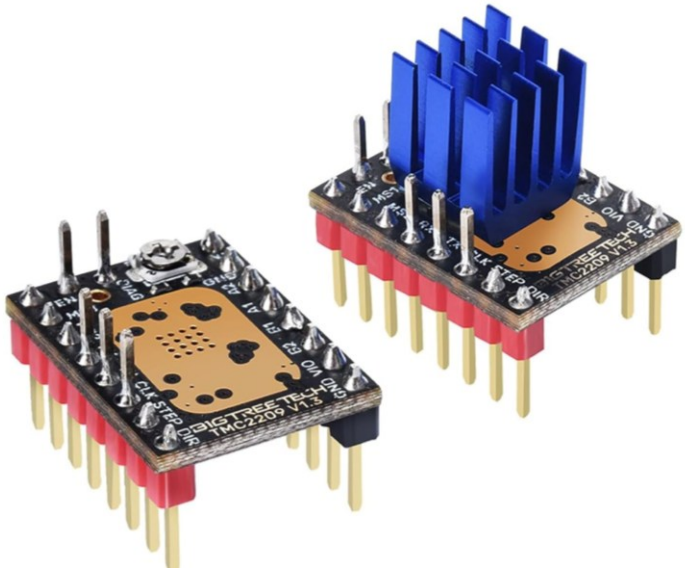


Figure 17: TMC2209 V1.3 Stepper Driver

The filament length encoder which was used for the test bench gives an output signal of pulses which are counted by the Microcontroller. The resolution of the sensor can be adjusted by twisting a potentiometer.

We use setting 2 which creates a signal of 6.421 pulses per mm.

	Rotary switch position	1	2	3	4
1.75mm version	pulses/mm	102.7	6.421	0.803	N/A
	mm/pulse	0.010	0.156	1.246	N/A

Figure 18: Filament Length Encoder Resolution

1.2.2 Pinout Microcontroller

All the mentioned components first were assembled on a breadboard to test the circuits.

To make the use of the test bench easier, a perfboard was soldered with connectors and wires for all the electrical components. A more beautiful solution would be to design and manufacture a PCB, but that would be an optimization for a following project. The ESP32 power supply is provided by a USB Kabel which is necessary to send the data from the microcontroller to the GUI where the measurement will be printed and shown in graphs.

Pins that can be accessed via connectors from outside the perfboard:

GPIO 4	NTC Pin (Analog-Input)
GPIO 5	Filament Length Encoder Pin (Digital-Input)
GPIO 6	Heater Pin (PWM Output)
GPIO 7	Fan Pin (PWM Output)
GPIO 3	HX711 Clock Pin
GPIO 8	HX711 Data Pin
GND	Ground of all Components
3,3V Volt Pin	MOSFET Module ref. Pin, Fil. Length encoder

Pins that cannot be accessed via connectors from outside the perfboard (wired up internally):

3,3V Volt Pin	Power supply for: Fil. Length encoder, Stepper Driver ref. Pin,
GPIO 9	Stepper Driver DIR Pin
GPIO 10	Stepper Driver STEP Pin
GPIO 11	Stepper Driver EN Pin

1.3 Software

1.3.1 Software concept

The software concept of the test bench is to collect data on an embedded device (ESP32), to send the data via. USB cable to the serial port of a laptop or computer where a graphical user interface receives the data, shows the data in text boxes and a plot and saves them into an CSV-files, stored on the laptop/computer where the GUI runs. The control of the test bench and the sending of the parameters of a measurement is done by the GUI.

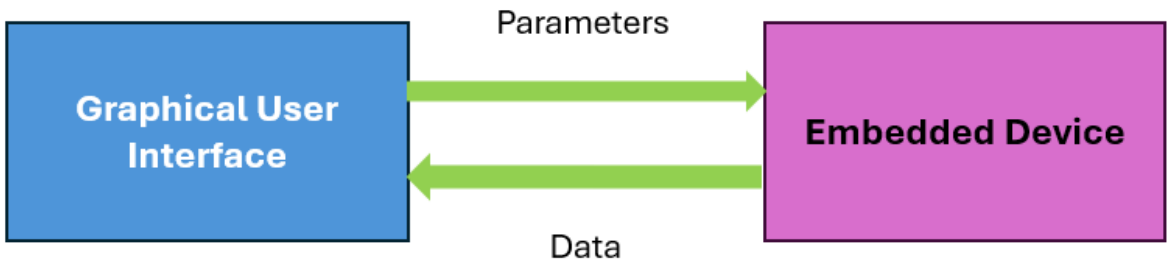


Figure 19: Software Concept

The software can be divided into two distinct parts.

1. The Embedded Software (C++ in Arduino Framework)
2. The Software for the GUI (Python)

1.3.2 Embedded Software

The purpose of the embedded software is to collect data from the sensors (temperature, fed filament length and the extrusion force) and to control the actuators (heater of the hotend and the stepper motor).

The software was implemented using object-oriented programming. Different classes were made to abstract data from the main program to make the logic for the main program as straightforward and clear as possible. All these classes were modularized in external header-files and cpp-files, to reach an obvious and good abstraction. Those files were included in the main.cpp file to use all the abstracted software.

External Modules:

- ExtruderStepper.h
- GUICommunication.h
- HotEnd.h
- LoadCell.h
- Rotary_Encoder.h

1.3.3 Main Program

Instead of using one endless loop in the main file to await the commands and parameters from the GUI the decision was made to use the multi-tasking method of freeRTOS. FreeRTOS is real time operating system for embedded systems.

The reason this method was used is because, if all the different functions were called one after the other, this could lead to timing problems which would result in vibrations from the stepper motor, or the defined measuring frequency were not possible anymore.

So, the main loop was subdivided into different tasks, so that every single task has its own endless loop. The whole scheduling of the task is done by the Scheduler of freeRTOS.

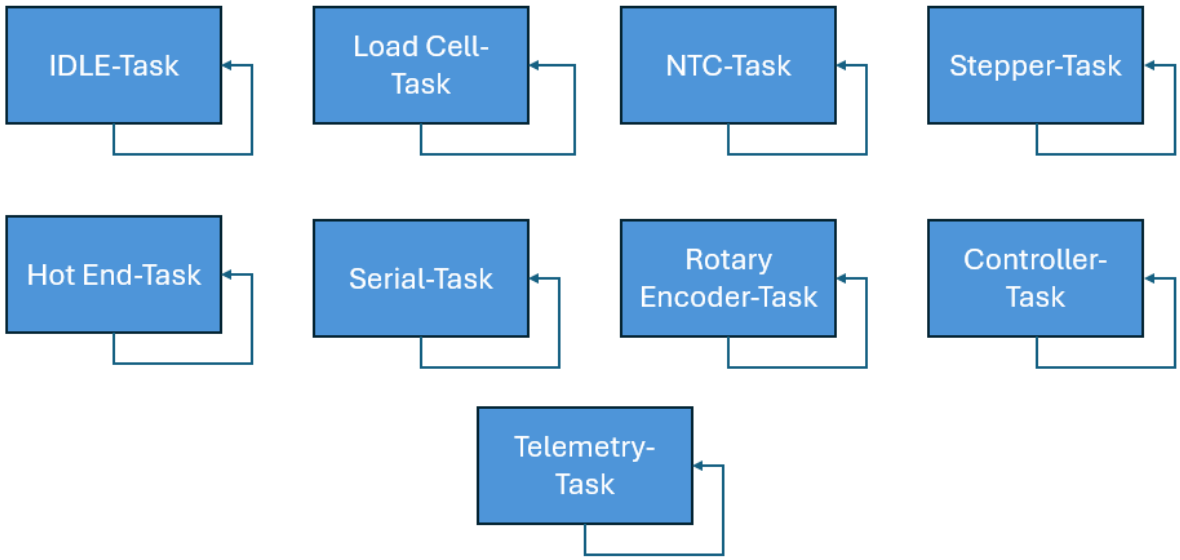


Figure 20: Multitasking

1.3.4 The Graphical User Interface

The GUI was implemented in python using the library tkinter and matplotlib. The purpose of the GUI is to visualize the incoming data (temperature, force and filament slip) in form of numbers and graphs and to start a measurement with different parameters.

The parameters are:

- Temperature in °C
- Feed rate in mm/s
- Feed length in mm
- Flow rate in mm^3/s (is calculated by the GUI)

It is also possible to tare the load cell. This would be important if the hotend would be changed.

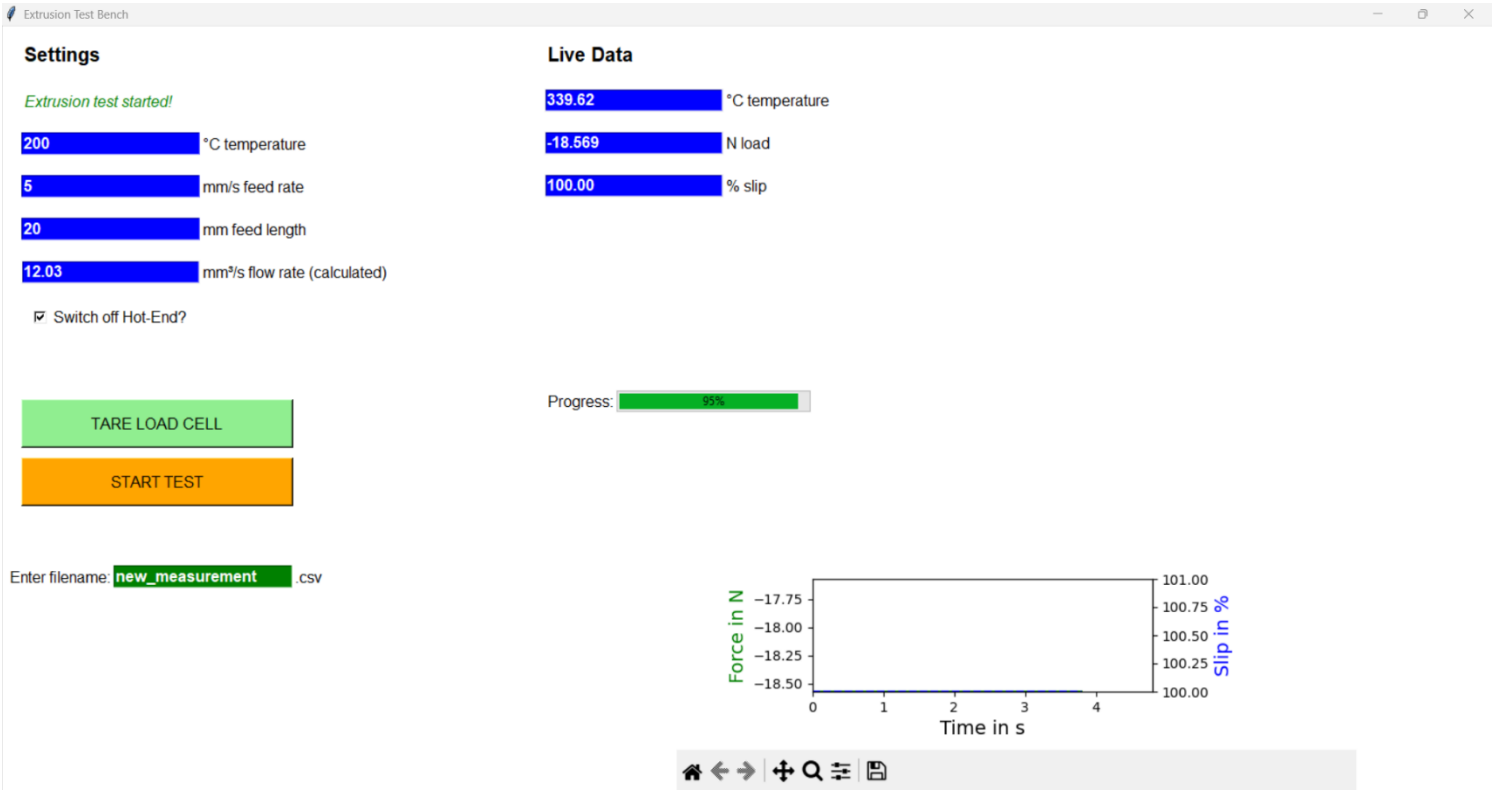


Figure 21: Graphical User Interface

More detailed documentation of the software will be provided in a separate document in this repository, as it would exceed the scope of this technical project documentation.