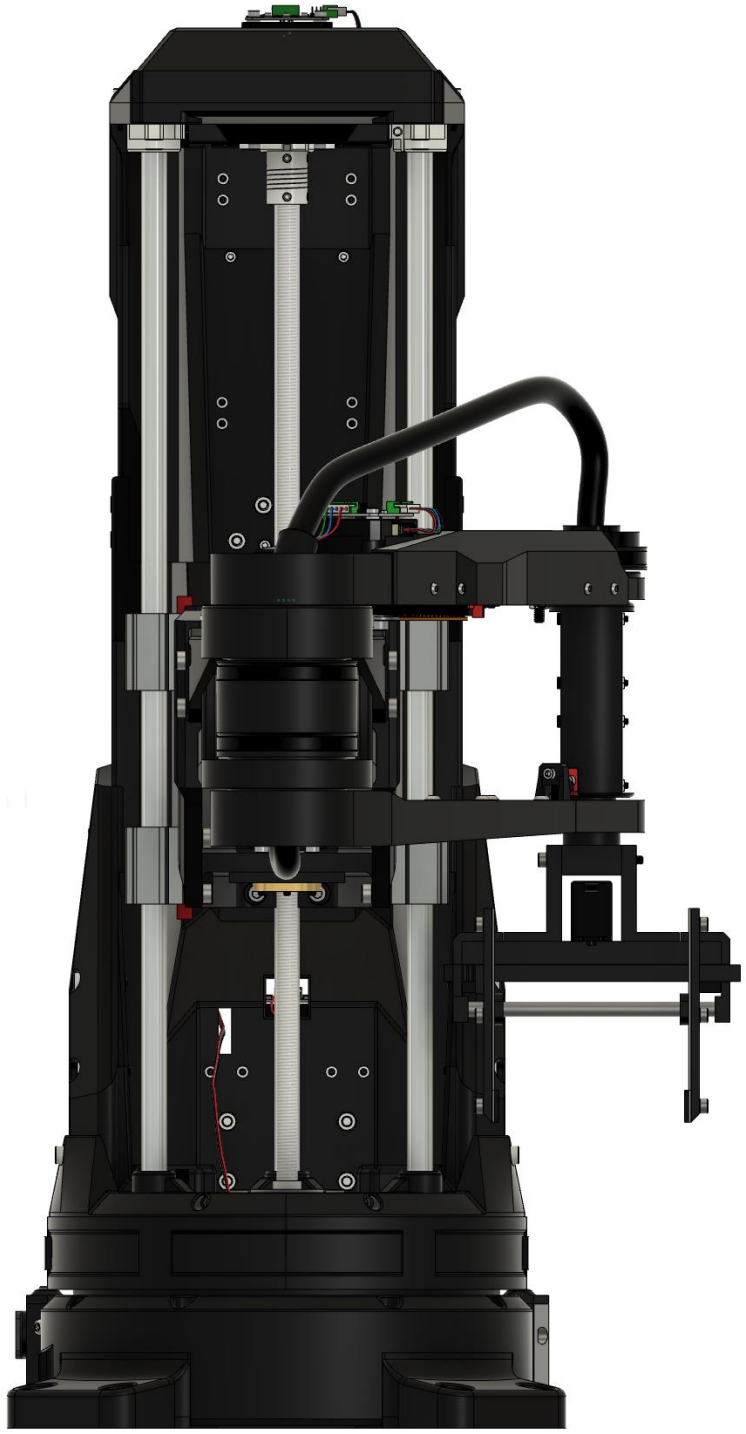


BIO SCARA MANUAL

March 2024

TABLE OF CONTENTS

Proposal – p.3
Parts List – p.14
3D Printing – p.16
Joints – p.75
Actuators – p.82
Endstops – p.87
Assembly – p.92
Electronics – p.202
MKS Gen L2.1 setup – p.204
Raspberry setup – p.209
MKS Servo 42C – p.212
Schematic – p.214
Software – p.226

PROPOSAL

CONTROL BOARDS

The robots consists of two boards:

- MKS GEN L2.1 handling low-level control of actuators and sensors
- Raspberry Pi 4

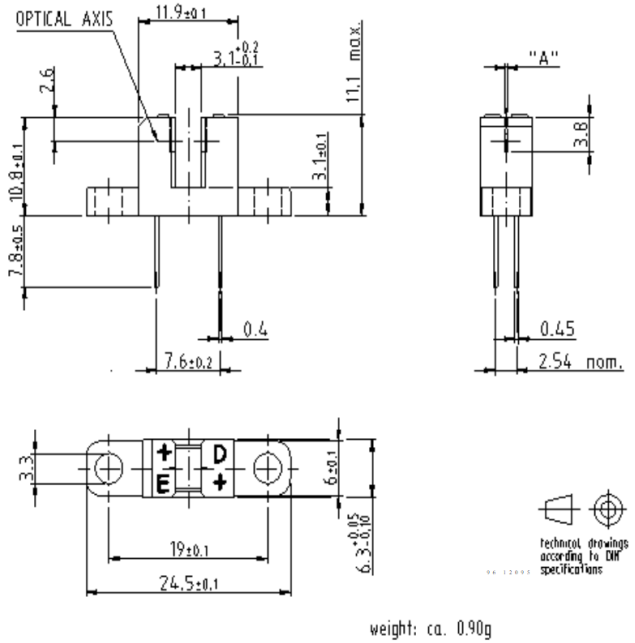
The reason for dividing the control system into two parts was to have a dedicated motherboard facilitating all the necessary I/O pins for all the peripherals. This board would serve as a slave and complete all received commands while the Raspberry would serve as the master where the scripts can be easily adapted without interfering with the low-level control scripts. This was an attempt to make the robot more user friendly since the programming on the robot revolves around some basic python scripts.

The MKS Gen L2.1 is a 3d printer motherboard and is a combination of Ramps 1.4 with ATMEGA2560. One advantage of using this board was the available open-source software Marlin. With some changes, it can be used to control a robot rather than a 3d printer.

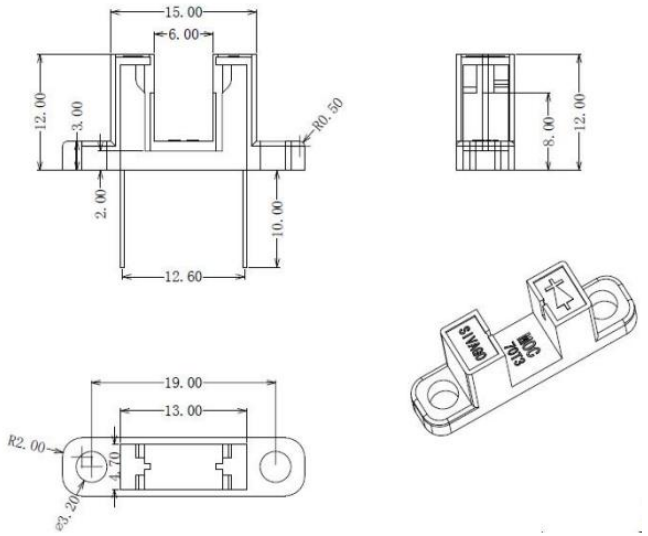
Even though Marlin comes with some built in features, a single-board system should be considered. This custom board would have to facilitate the connection of stepper drivers (in this case MKS Servo42C and other peripherals such as the endstop sensors.

ENDSTOPS

For the endstops sensors, photointerrupter sensors have been chosen due to their low price as well as the fact that they do not require physical contact allowing for full 360° joint rotation. The specific model used in the prototype is TCST2103 with a 3.1mm gap, however a different model can be used such as MOC70T3 which has a 6mm gap and can be used with the same sensor mount as the TCST2103.



TCST2103



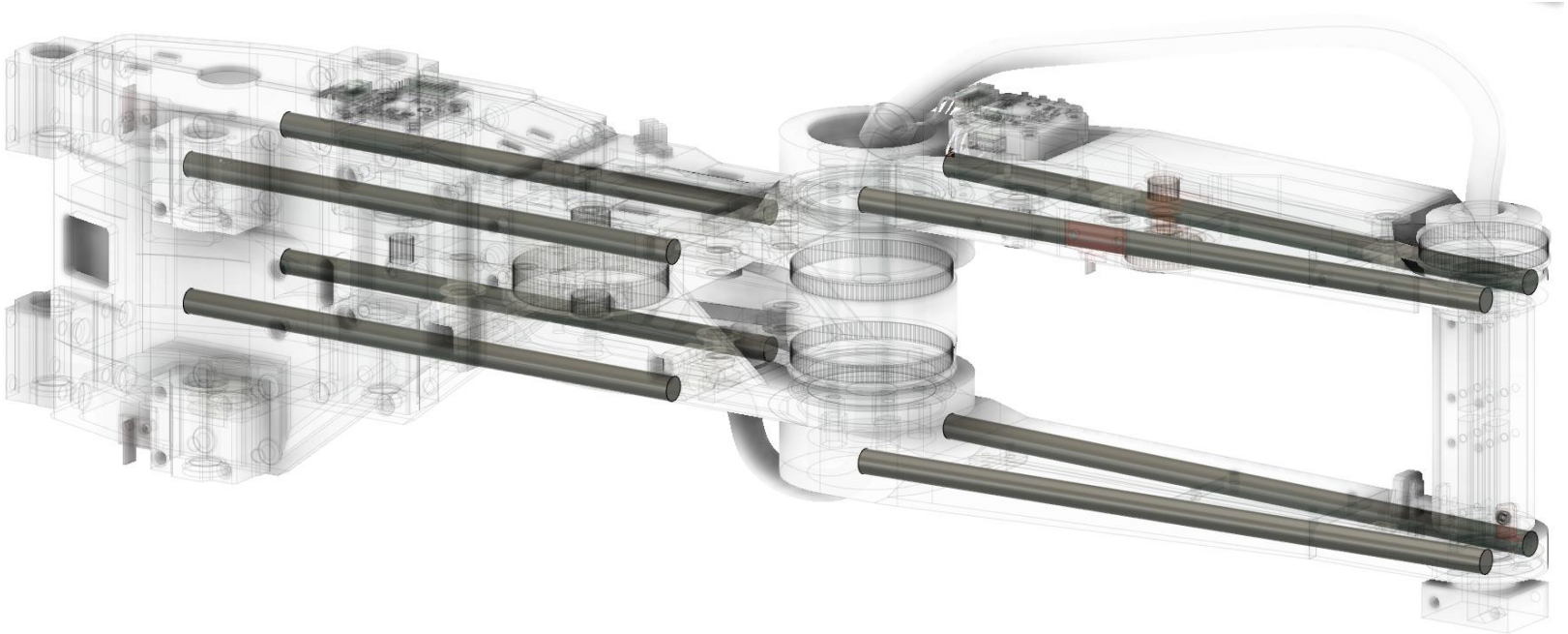
MOC70T3

STRUCTURE REINFORCEMENT

The overall mechanical structure should be reinforced with light-weight rigid parts (such as metal or carbon-fiber tubes). This should reduce the arm's flexibility which should also reduce the end-effector's vibrations. As an example, consider the same arm structure but with holes parallel to the arm. These would be designed for the insertion of rods of tubes.

These would improve the mechanical properties of the arm in the weakest point.

Of course, the flexibility will always remain in the parts where the arm links are connected through joints (for example due to the reliance on timing belts) but it should be to a smaller extent.

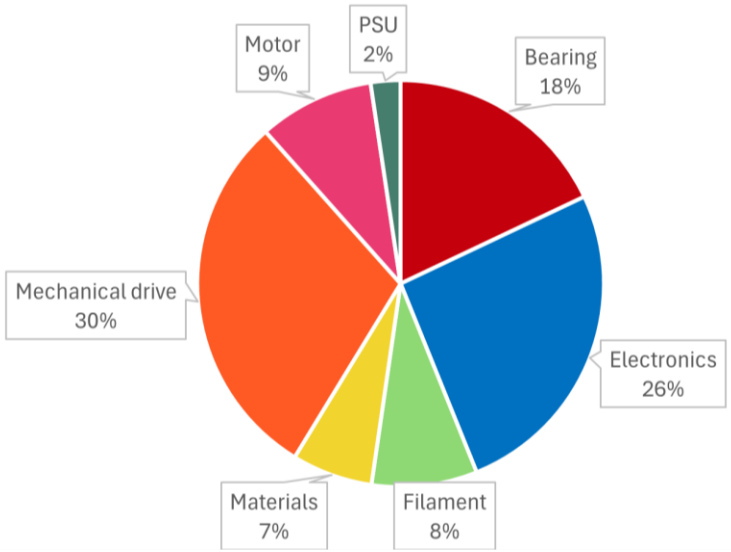


COST REDUCTION

While the main goal was to use off-the-shelf components, the cost of the project can be lowered even further. By sourcing components solely from websites like Aliexpress rather than Amazon, the cost can be lowered significantly. As a general rule, it was assumed that items from the former website are 70% of the cost of the latter (the cost includes both the item's cost and the shipping cost). Additionally, a smaller power supply was chosen as the motor's were driven with an adaptive current, resulting in requiring much less power than initially expected. By choosing a 100W power supply and utilizing an ESP32, the cost estimate came out to around 3200dkk (460 usd). The biggest downside of choosing this sourcing method is the big lead time (around one month).

NOTE:

This cost does not include parts like screws, nuts, washers and heat inserts.

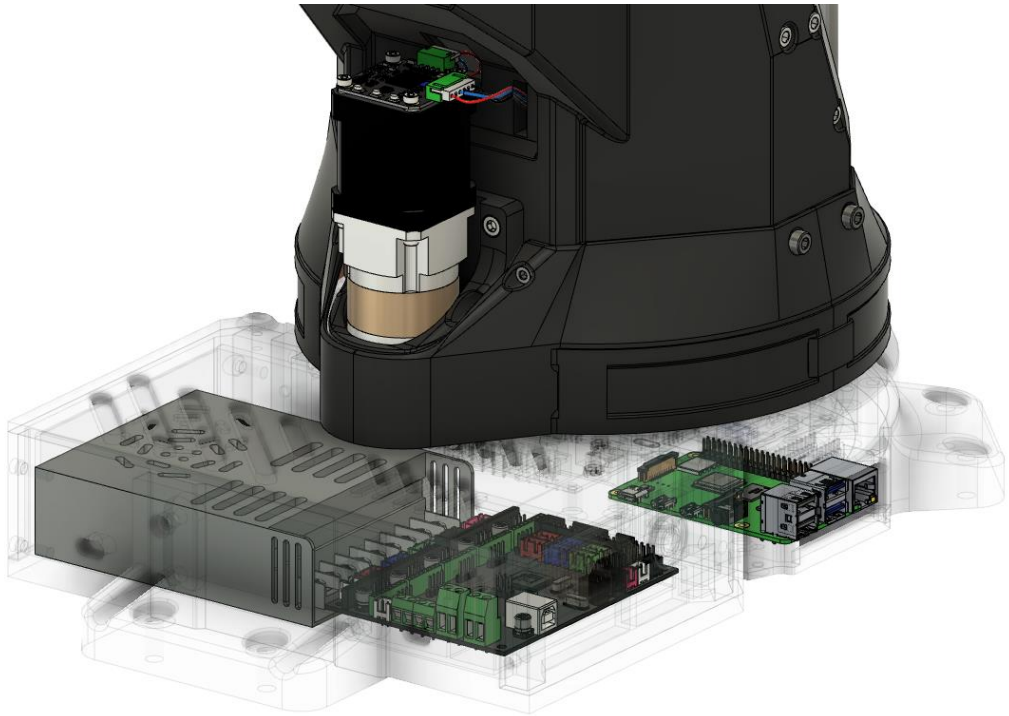


POWER SUPPLY

MKS Servo42C stepper drivers have an adaptive current functionality. This means that the current provided to the steppers is kept low during normal motor operations and increases when the positioning error of the shaft increases and has to be accounted for.

The robot uses no more than 30% of the maximum possible output power (400W), therefore it is possible to use a smaller and cheaper power supply which also results in a more compact design. A 100W power supply is enough to power all actuators and electronics however one should consider the specific application in which the robot will be used. This includes considerations on what end-effector will be used and if other peripherals will be powered from the robot's power supply.

Using a more compact power supply would result in more available space left for the electronics where for example, a control board could be placed to improve the cable management of joint 1.



CABLE MANAGEMENT

Even though the joint can be rotated 360° it is important to note that all the moving joints are connected in one fixed place – the MKS GEN L2.1 motherboard. Therefore it is not advised to rotate joint 1 more than ±180. The robot should be homed by going in the opposite direction of the original path.

If joint 1 and joint 4 continue to rotate, they will pull a connectors out of the motherboard. In the worst case scenario, the cables get severed and will have to be repaired.

A different danger is the wires tangling around the sensor trigger of joint 1. Therefore a box to contain all the wires as well as to cover the electronics has been deployed. It should be mentioned however that this is a short term solution and the overall cable management of wires in the robot base have a lot of space for improvement.

GRIPPER

The robot can be equipped with various gripper types or devices. In the initial application, the gripper is responsible for pick and place tasks of well plates and that is why the gripper tries to mimic the standard grippers used in the bio industry however while utilizing only off the shelf components.

Standardization of well plates follows the ANSI/SLAS Microplate Standards. For this project, the main aspects of these standards included the ANSI/SLAS 12004 for footprint dimensions, ANSI/SLAS 22004 for height dimensions, and ANSI/SLAS 42004 for well positions. Such standardization ensures compatibility with a range of automated laboratory equipment. The standardized footprint for microplates is 127.76 mm by 85.48 mm, with a tolerance of ± 0.5 mm. It is important to note that while the footprint and well positions of these plates are standardized, the height dimension and number of wells can vary. This variation comes from different volumes the plates are designed to hold and the specific designs of the wells

By building a prototype, It was determined that having a strictly 3D printed gripper is not feasible. Various materials have been used, ranging from a typical PLA to much stronger materials like PA CF. By doing several iterations and building multiple prototypes, it has been concluded that building a gripper made out of both plastic and metal is the suitable solution.

Note:

The gripper design still has a lot of room for improvements, with the main focus on making the gripper more compact, rigid as well as to increase its strength.

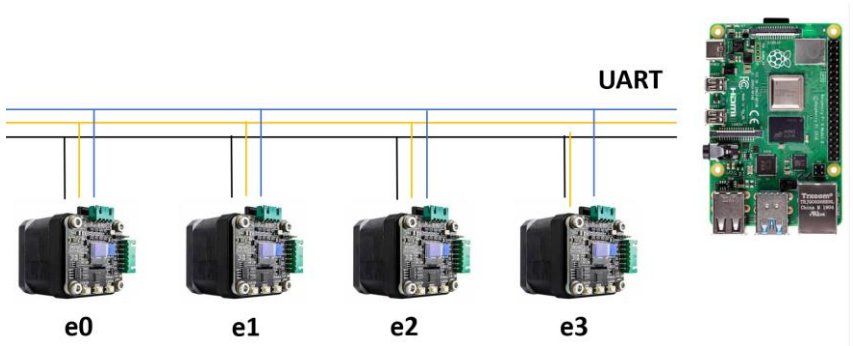
COLLISION DETECTION

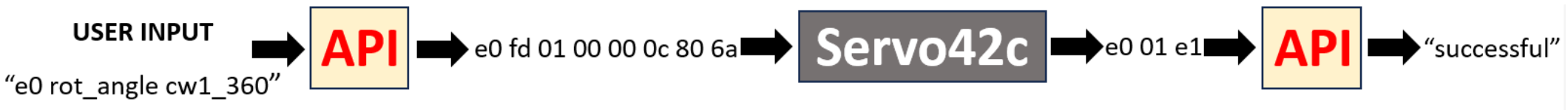
A safety feature which is worth considering for implementation is a collision detection. This can be achieved for example through the implementation of a camera or an additional sensor.
The MKS Servo42C already has a built in safety feature in the form of stall protection. You can find some information about it in the MKS Servo42C manual. This feature cuts of the motor's power and releases the shaft whenever a shaft stall has been detected.
This feature unfortunately has some downsides. The main disadvantages are no feedback to the MKS motherboard or Raspberry Pi. This is a problem since only the specific joint which has encountered a collision will be stopped while the other joints will continue working since the control boards have no information on the collision.
Another issue is the reliability of the stall detection – there can occur false triggers as well as no triggers (although rarely) when an actual collision has happened. Additionally, for joint 3 and 4 there is a risk of the motor continuing to rotate while the joint has been stopped – this is due to the GT2 timing belt slipping on the gears.

That is why it is good practice to consider other ways of collision detection.
For example, vibrations can be measured to evaluate whether a collision has happened.
To measure vibrations an MPU6050 module can be used. Due to the presence of vibrations, a static threshold might not the right decision. A dynamic threshold could be made based on the known motion profile

POSITION FEEDBACK

Throughout this project, a different control mode has been tested. The MKS Servo42C allow for direct motor control through UART communication. The MKS Servo42C are capable of being standalone motor drivers by independently generating impulses for the steppers. This means that when given a control command, these drivers can generate the appropriate impulses for the motor. This control method would be the ideal solution since it would give feedback signal from the magnetic encoders directly to the Raspberry. Not only would that mean more advanced control functions but it would also make the electronic connections more streamline, as all the motors can be connected together via UART and the power can be connected in a cascade.





Unfortunately it was determined after diligent testing that the current state of the firmware heavily limits the performance and control capabilities. The biggest issue is that the motor and driver still act as an external closed-loop and in order to retrieve the encoder data, a query has to be made to the driver. In UART mode, there are two main motion commands:

- Move motor to an angle
- Continuously rotate motor until interrupted

As it turned out, querying for the encoder data while the motor is performing a motion to a defined angle, interrupts the current motion.

The other, continuous rotation mode, allows to access the encoder data without interrupting the motion, however the motion would then have to be stopped through the Raspberry Pi that is making decisions based on the encoder data in real time. The approach being employed is less than ideal, given that a Raspberry Pi is not intended for realtime control.

Moreover, it results in the loss of one of the major benefits of stepper motors, which is a dependable open-loop control system. This is because in the latter motion mode, there is no data available on how many pulses have been generated for the motor. The fact remains that this control method (when improved through future firmware up dates) has a lot of potential as it would allow to use a Raspberry Pi as the only control board since the low-level control would be achieved through the motor drivers (eliminating the problem of a Raspberry Pi's lack of a real time clock and not being designed for controlling low-level components in real time).

PARTS LIST

SCREWS, NUTS, WASHERS

M5

NUTS
87x M5 NUT

WASHERS
13x M5 WASHER

SCREWS
5x M5X100
1x M5X75
4x M5X50
2x M5X45
2x M5X40 FLAT HEAD
4x M5X40
3x M5X35 FLAT HEAD
3x M5X30 FLAT HEAD
6x M5X30
32x M5X20
44x M5X15
10x M5X10

M4

NUTS
44x M4 NUT

WASHERS
2x M4 WASHER

SCREWS
3x M4X70
4x M4X40
1x M4X30
4x M4X20
18x M4X15
38x M4X10
4x M4X8 FLAT HEAD
5x M4X8

M3

NUTS
33 M3 NUT

WASHERS
4 M3 WASHER

SCREWS
8x M3X30
4x M3X20
4x M3X10 FLAT HEAD
52x M3X10
29x M3X8
9x M3X6

M2

NUTS
33 M3 NUT

WASHERS
4 M3 WASHER

SCREWS
4x M2x6

3D PRINTING

3D PRINTER

Parts have been printed on a Bambulab P1S printer. This printer has a print volume of 25dm^3. This volume is required for some bigger parts of the robot base and joint 1.

FILAMENT

The parts have been manufactured from PLA and PETG. It has been estimated that around 4kg of filament is required (1kg of PLA and 3kg of PETG but this ratio depends on the user's choice of filament where the filament type could be either of those.

MODELS

STL files have been added A complete Fusion360 model can also be found on github

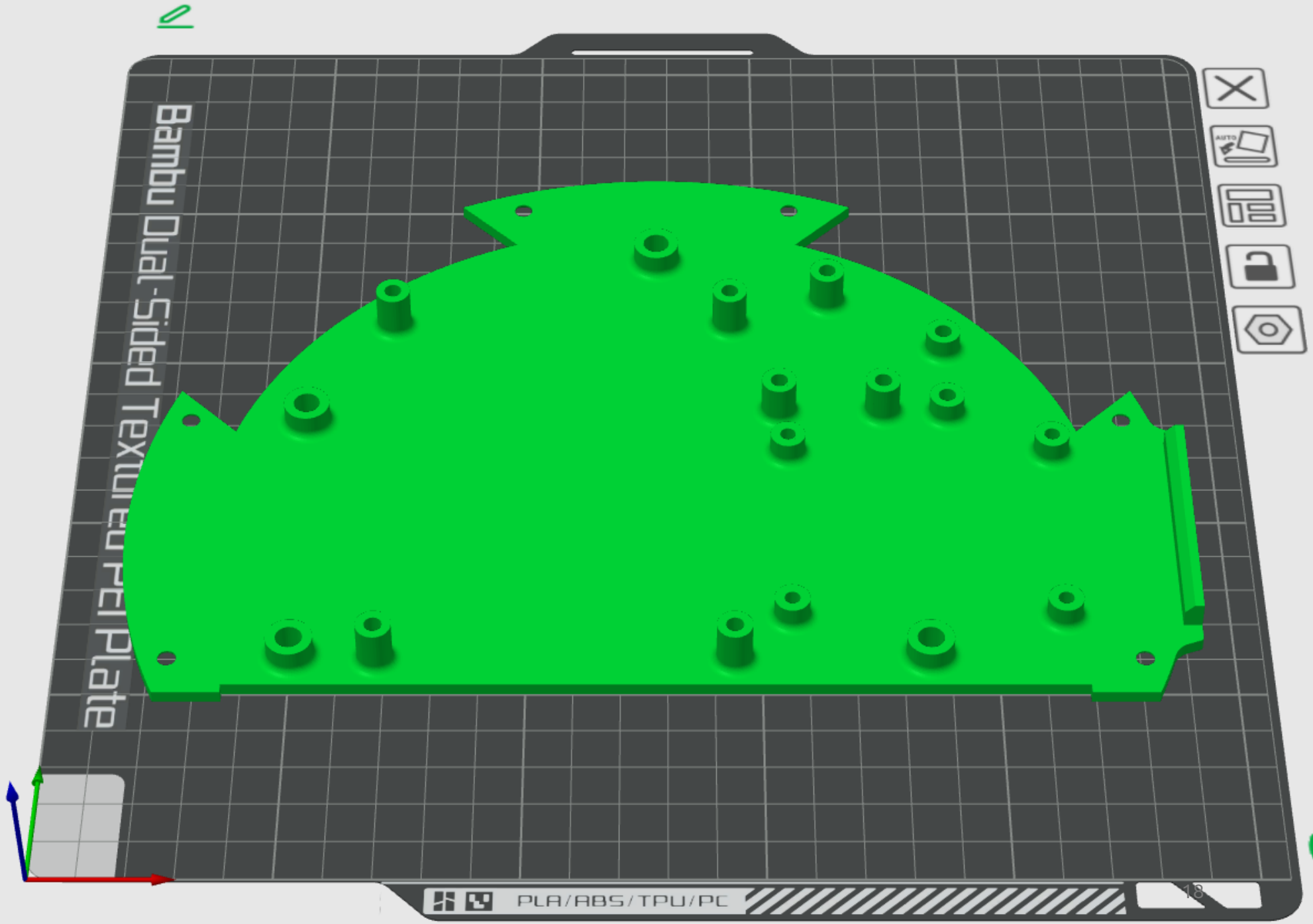
MANUFACTURING

The following slides illustrate the suggested print parameters for each model as well as the correct print orientation. Of course, multiple parts can be printed at once to reduce overall print time .

Bottom cover

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**



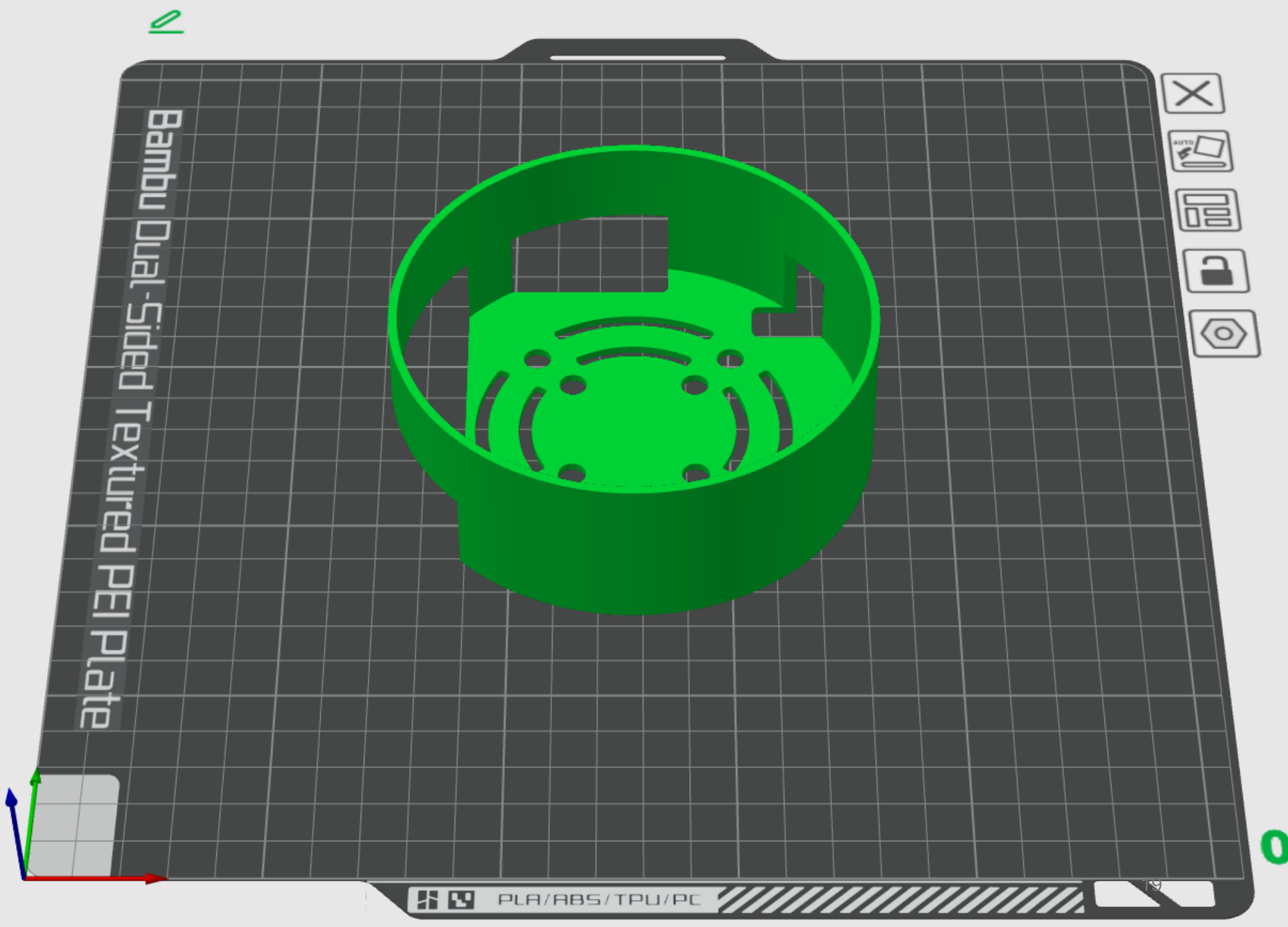
Cable box

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

Note:

Support is required

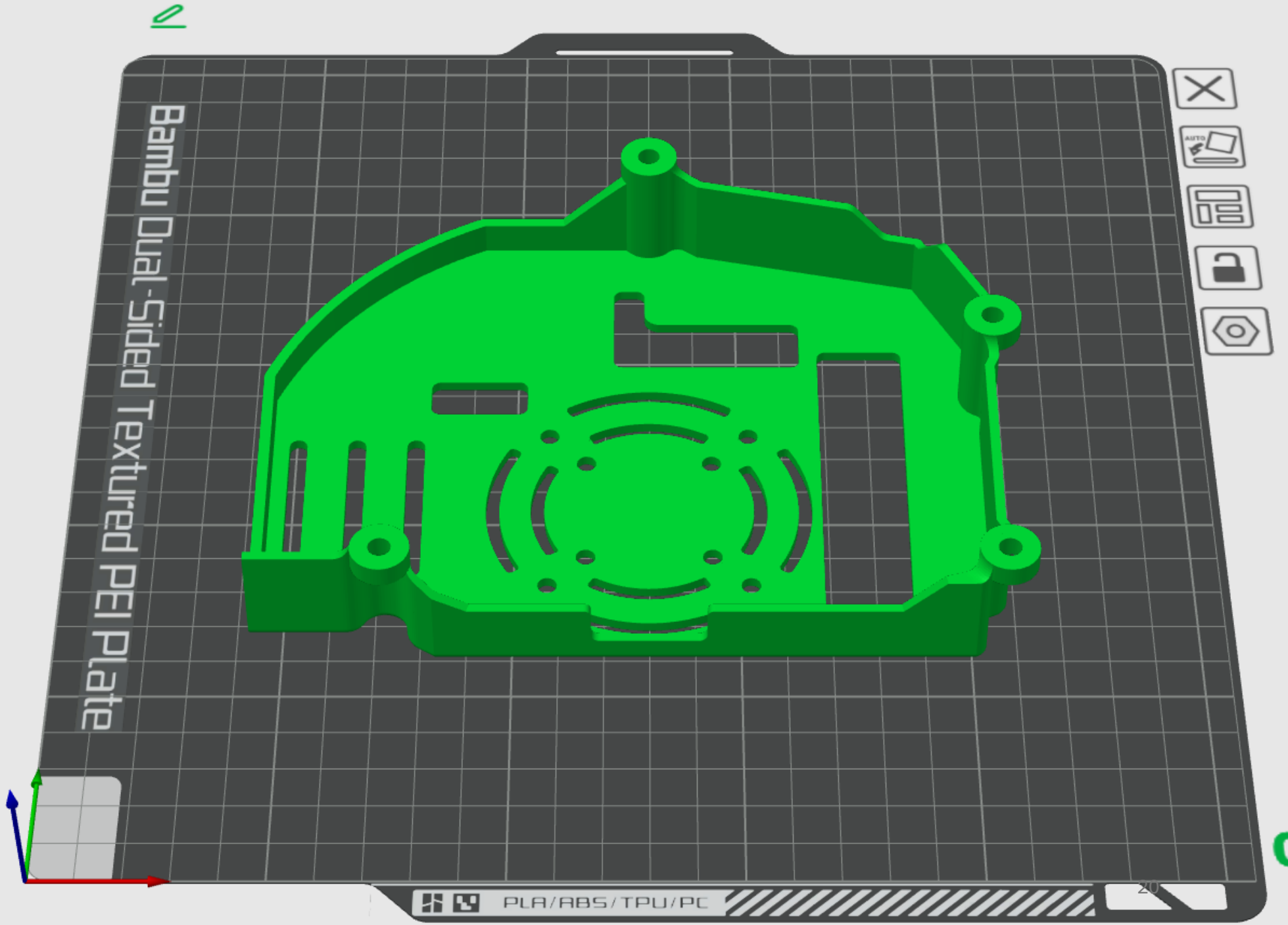


Electronics cover

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

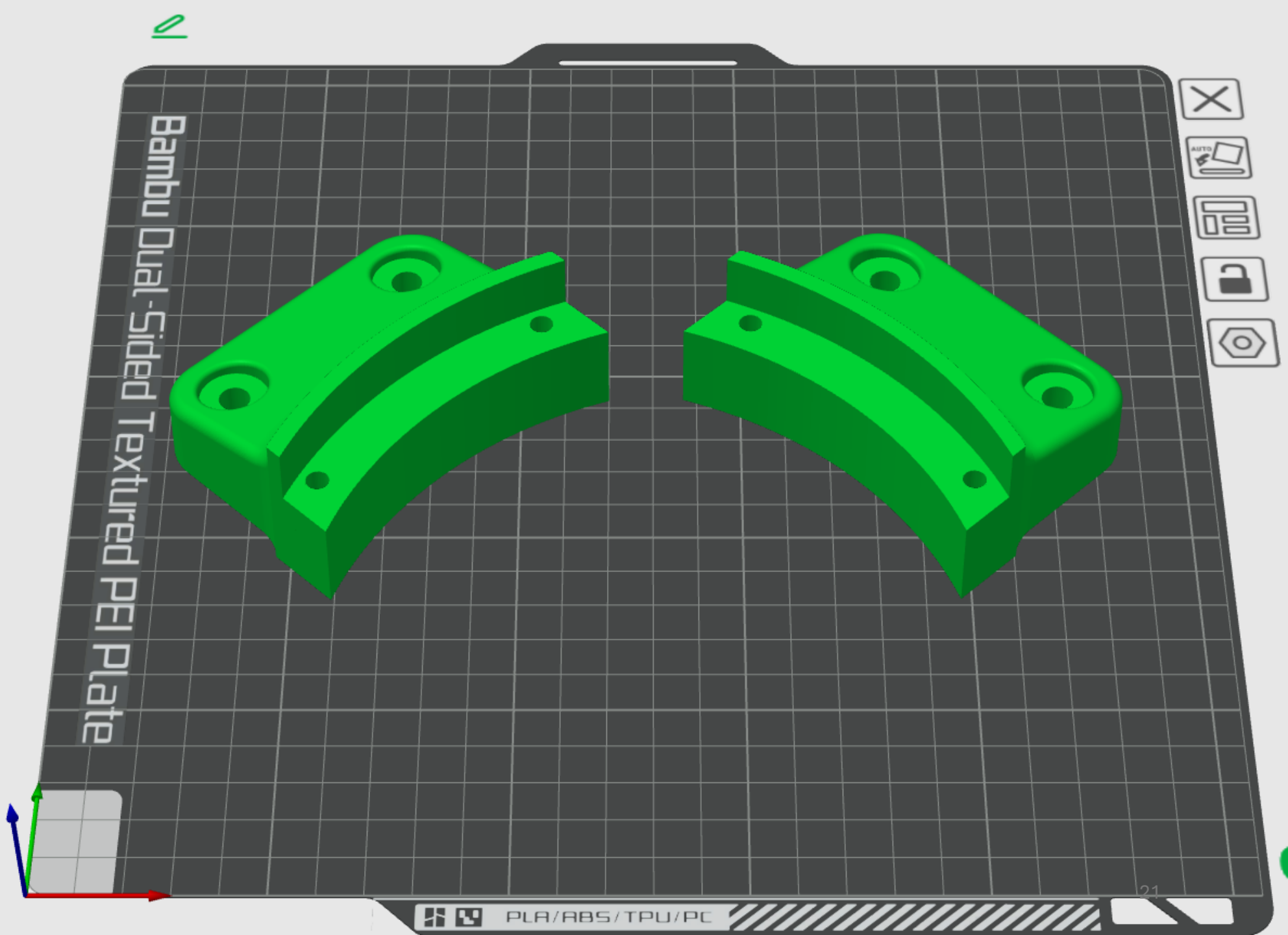
Note:
Support is required



Leg 1&2

Print parameters:

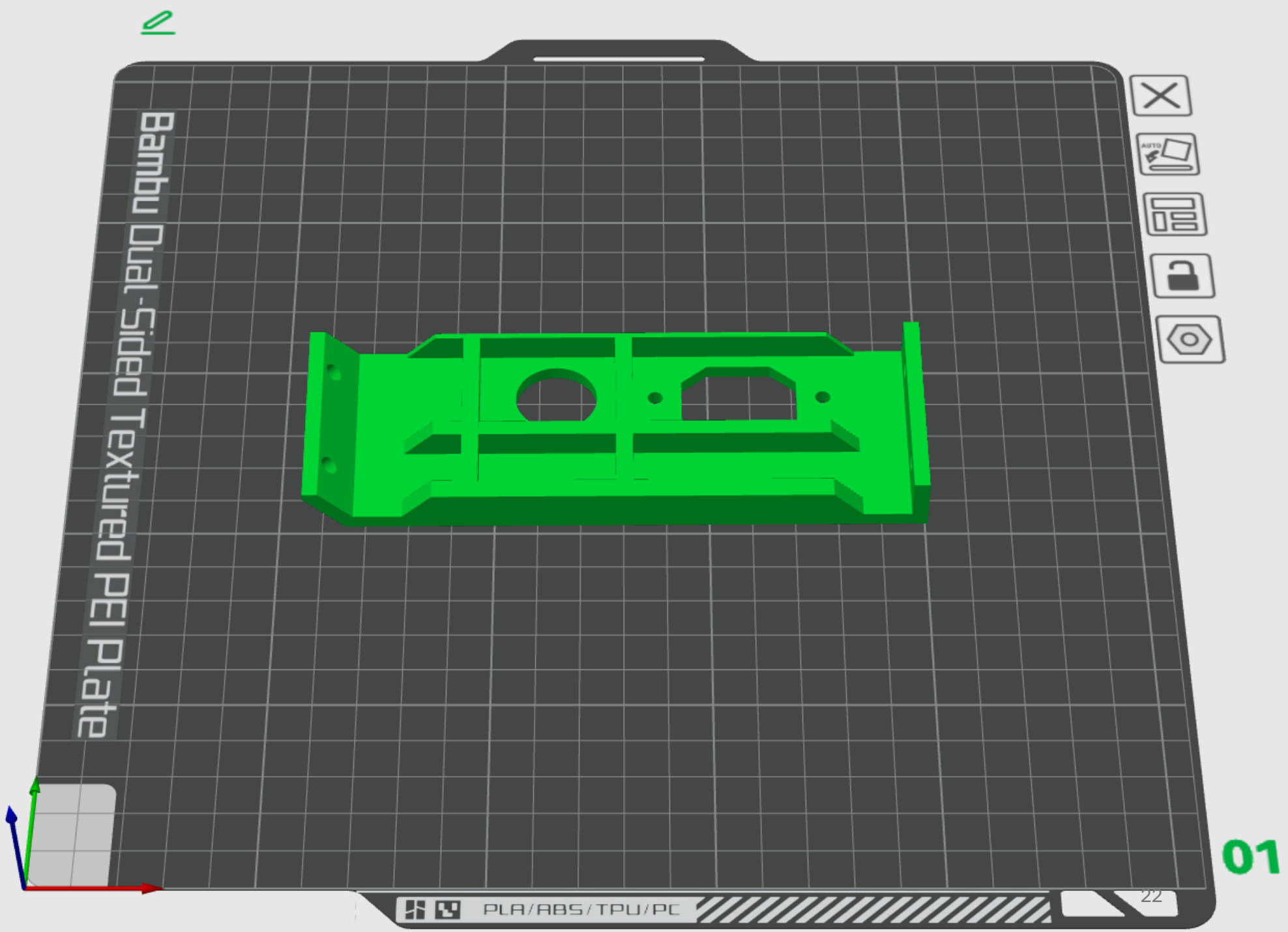
- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**



PSU front panel

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

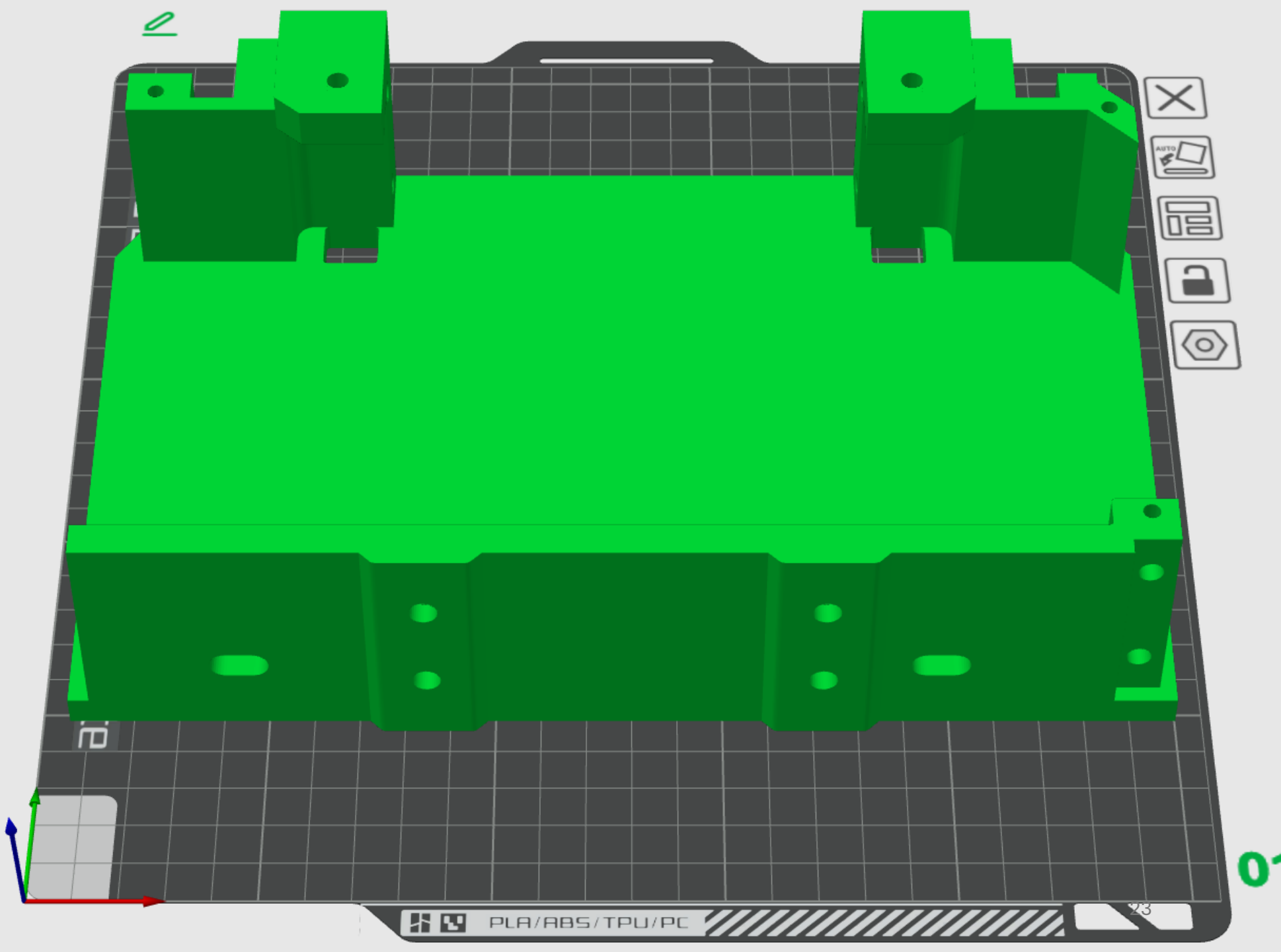


PSU box

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

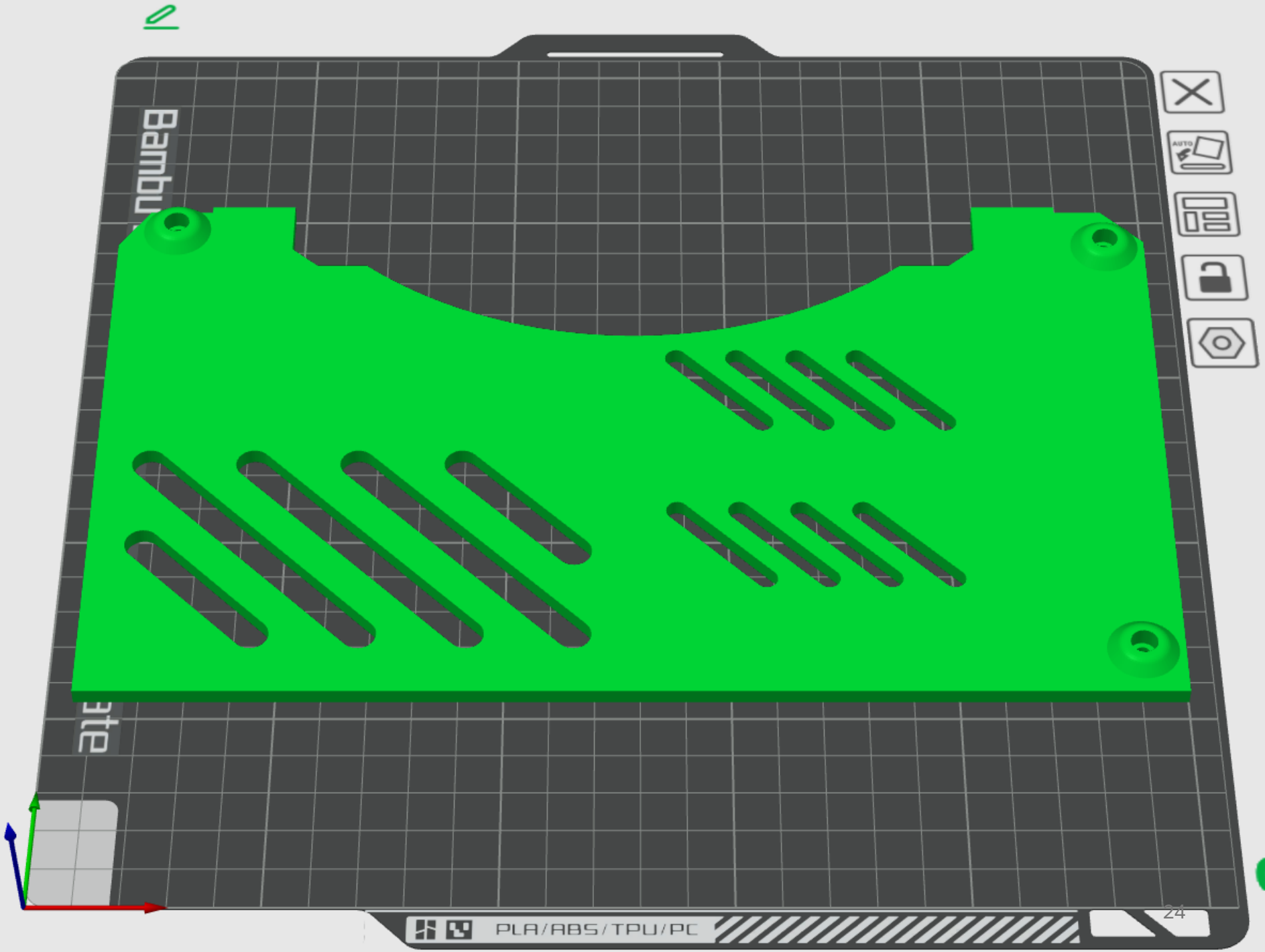
Note:
Support is required



PSU cover

Print parameters:

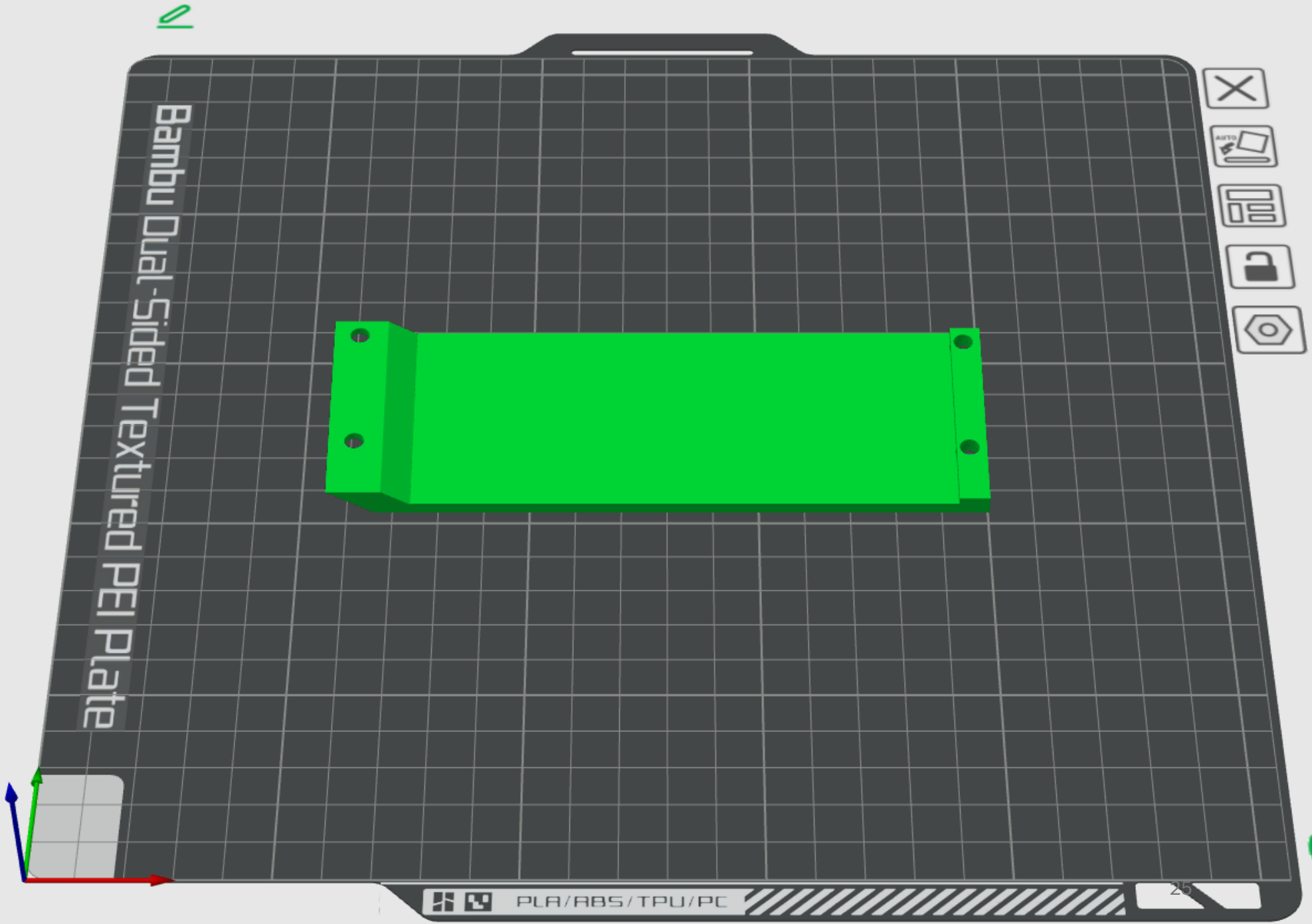
- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**



PSU rear wall

Print parameters:

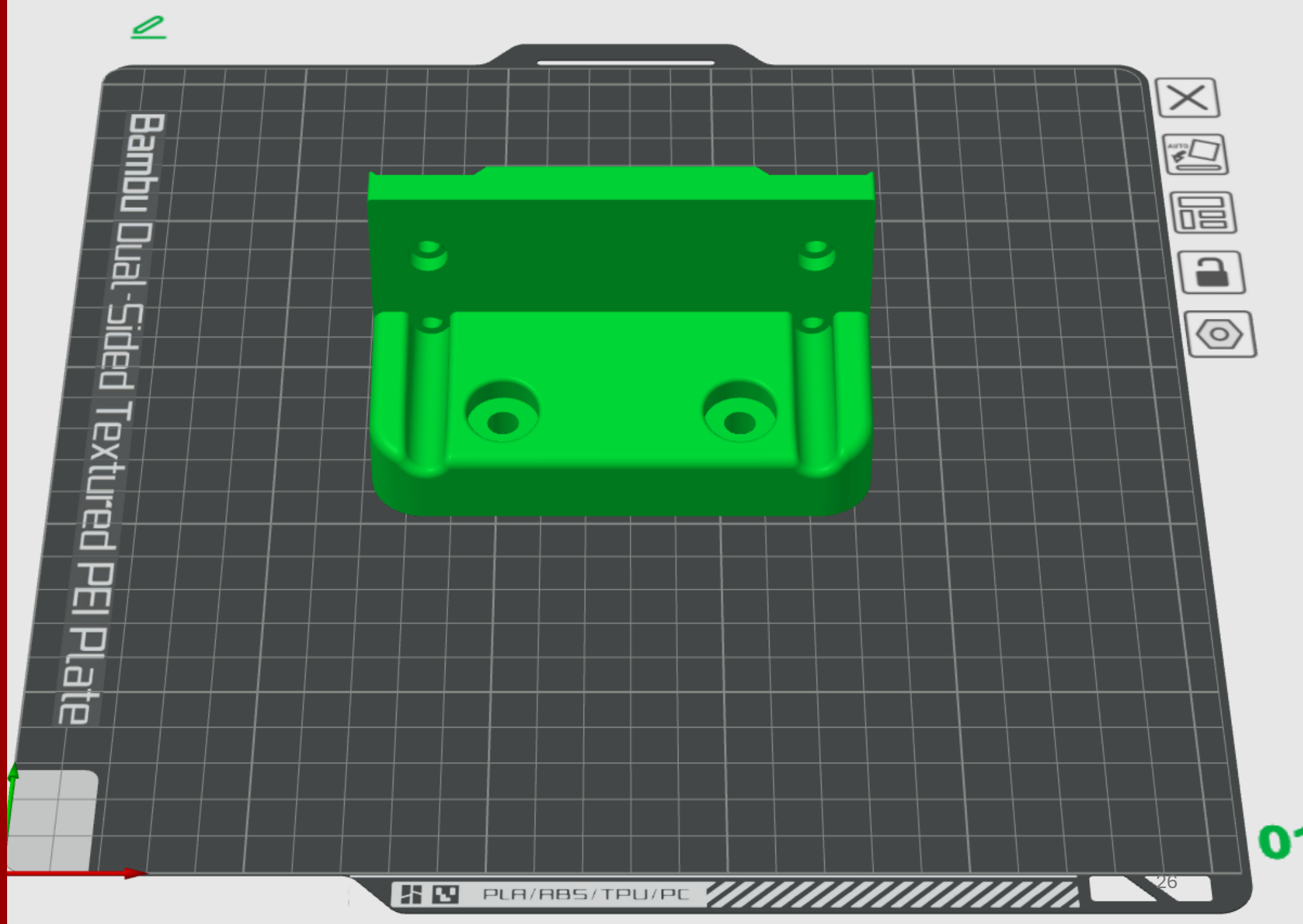
- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**



Rear leg

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**



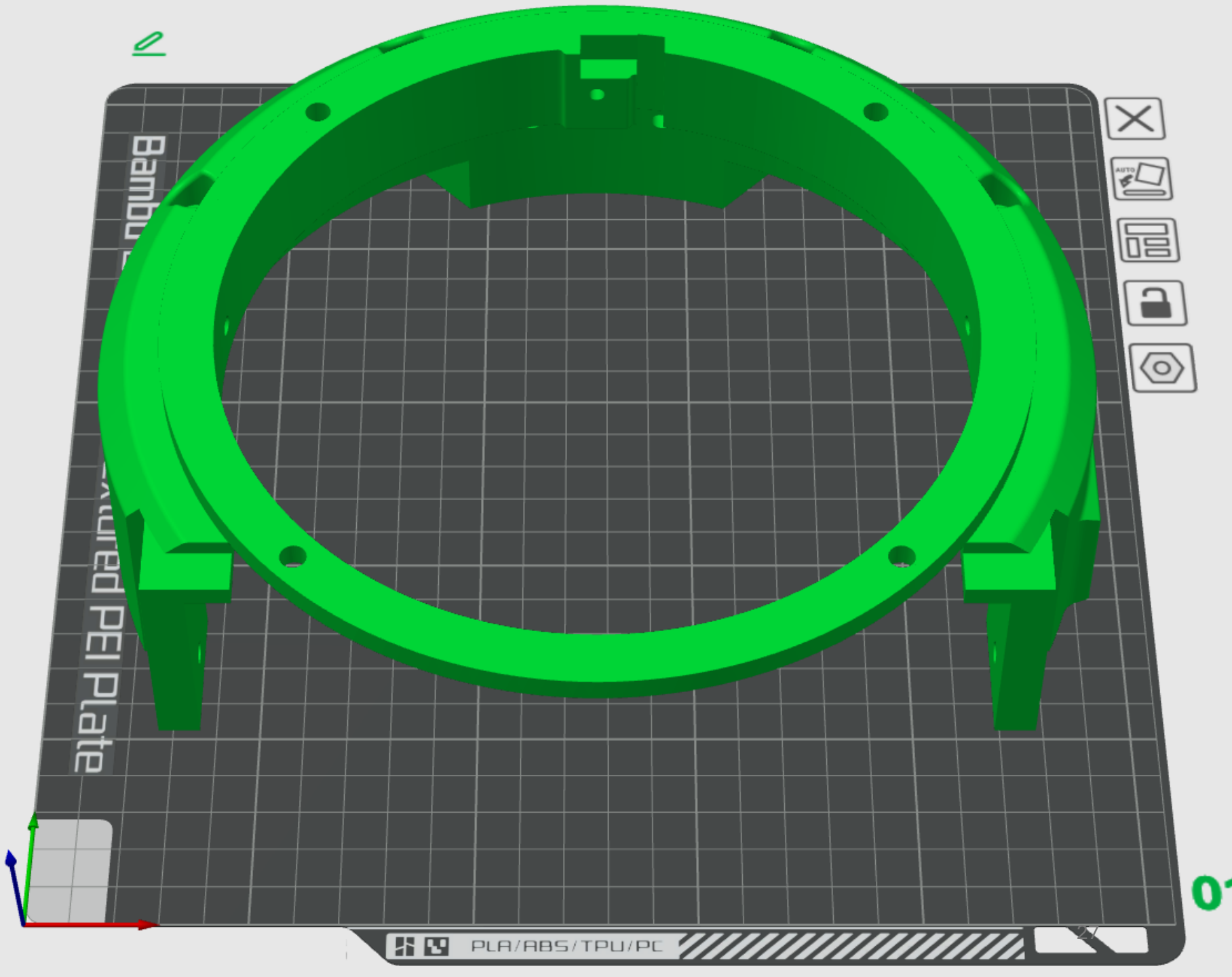
Robot base body

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

Note:

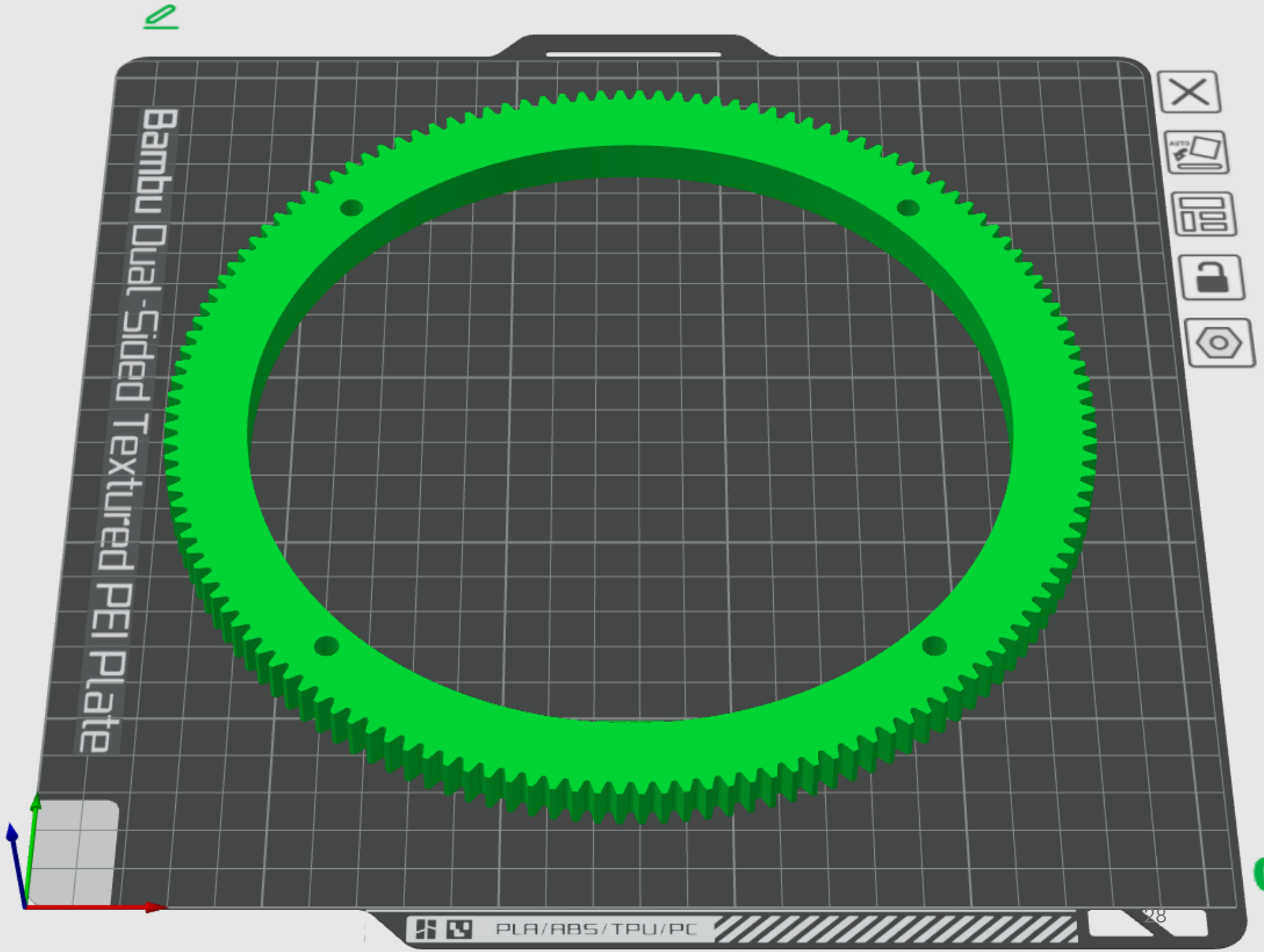
Support is required



Spur gear 140T

Print parameters:

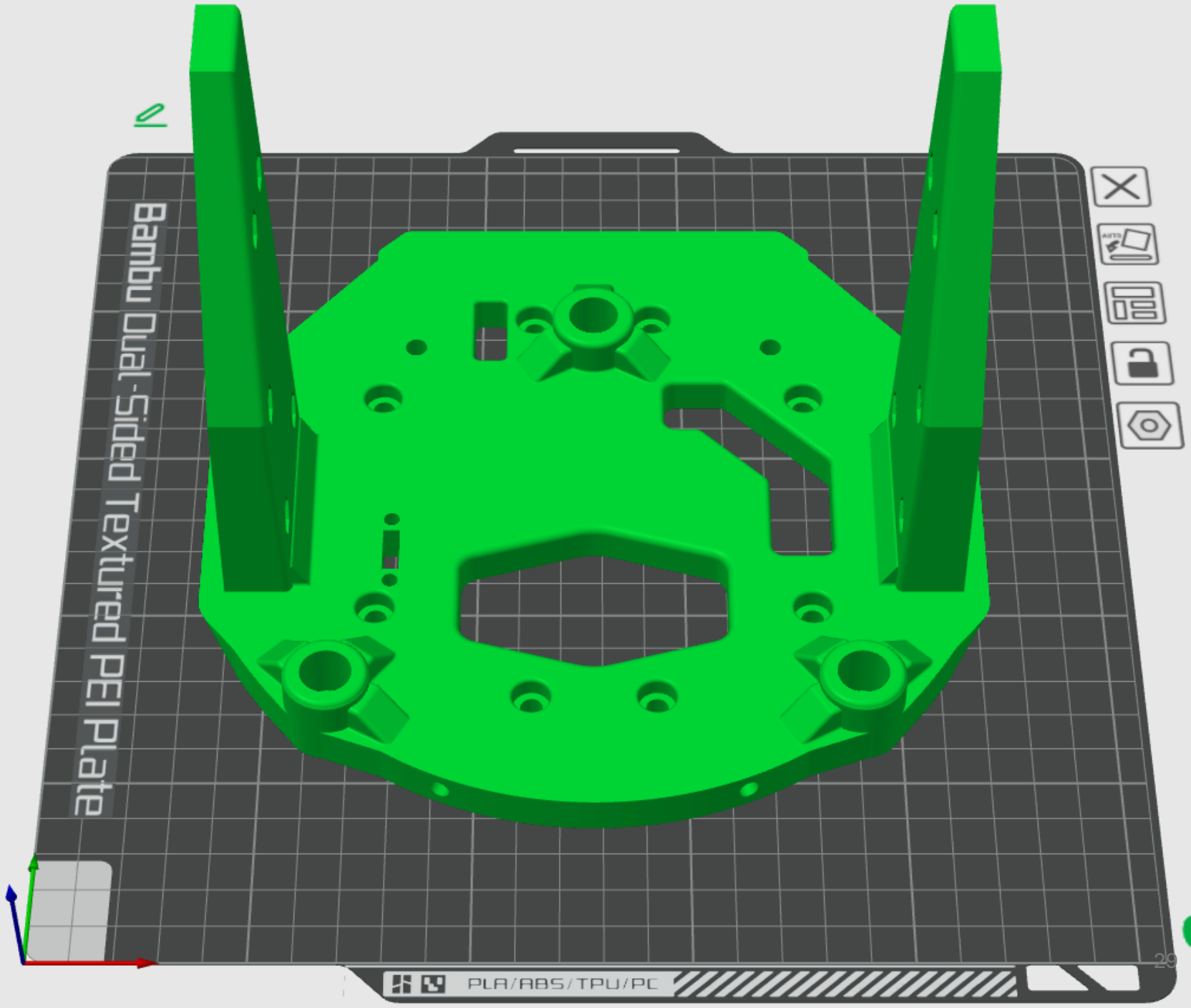
- **Material: PETG**
- **Layer height: 0.1mm**
- **Infill: 100%**
- **Wall loops: 4**



Shell mount

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

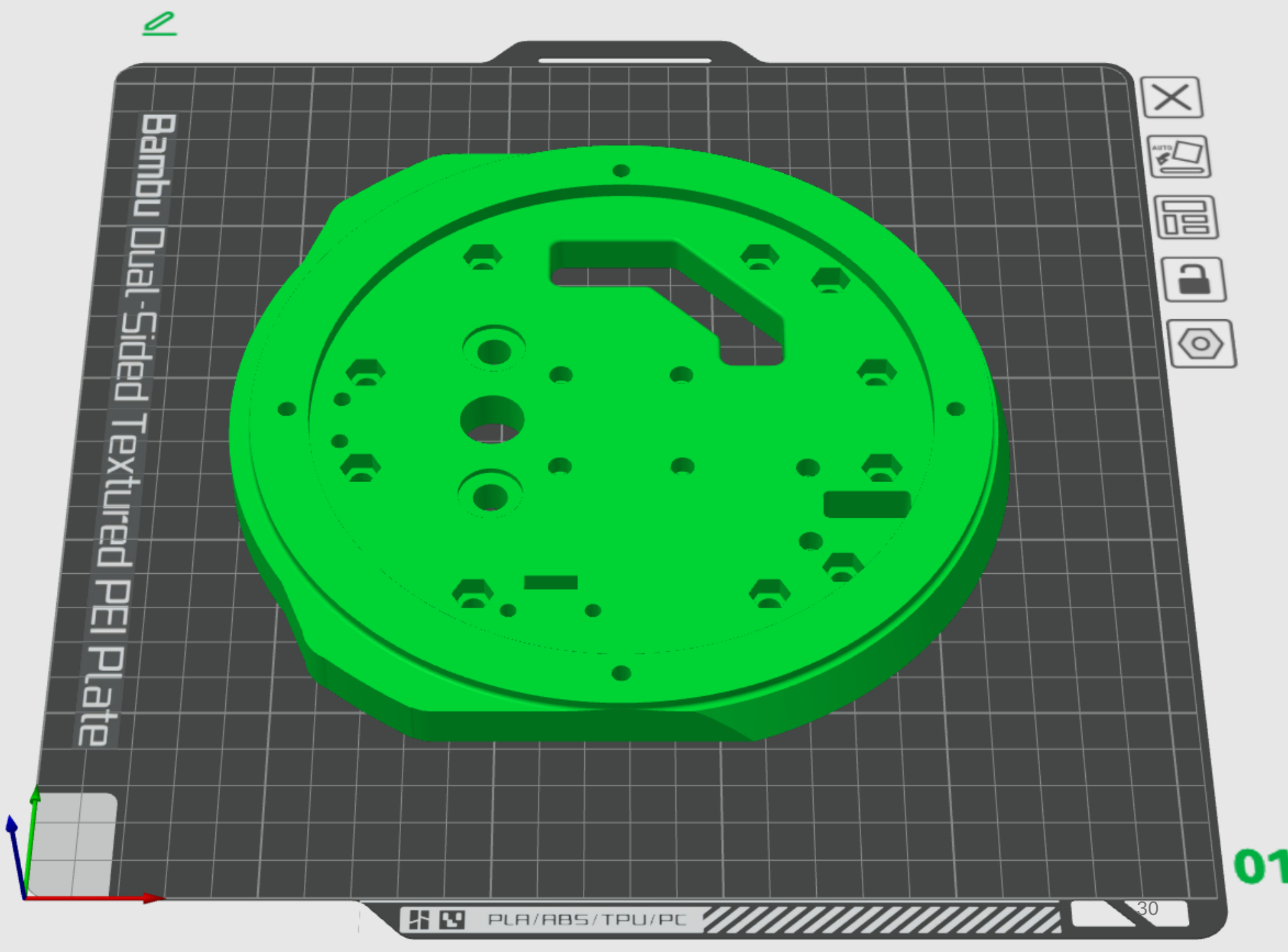


J1 base

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

Note:
Support is required

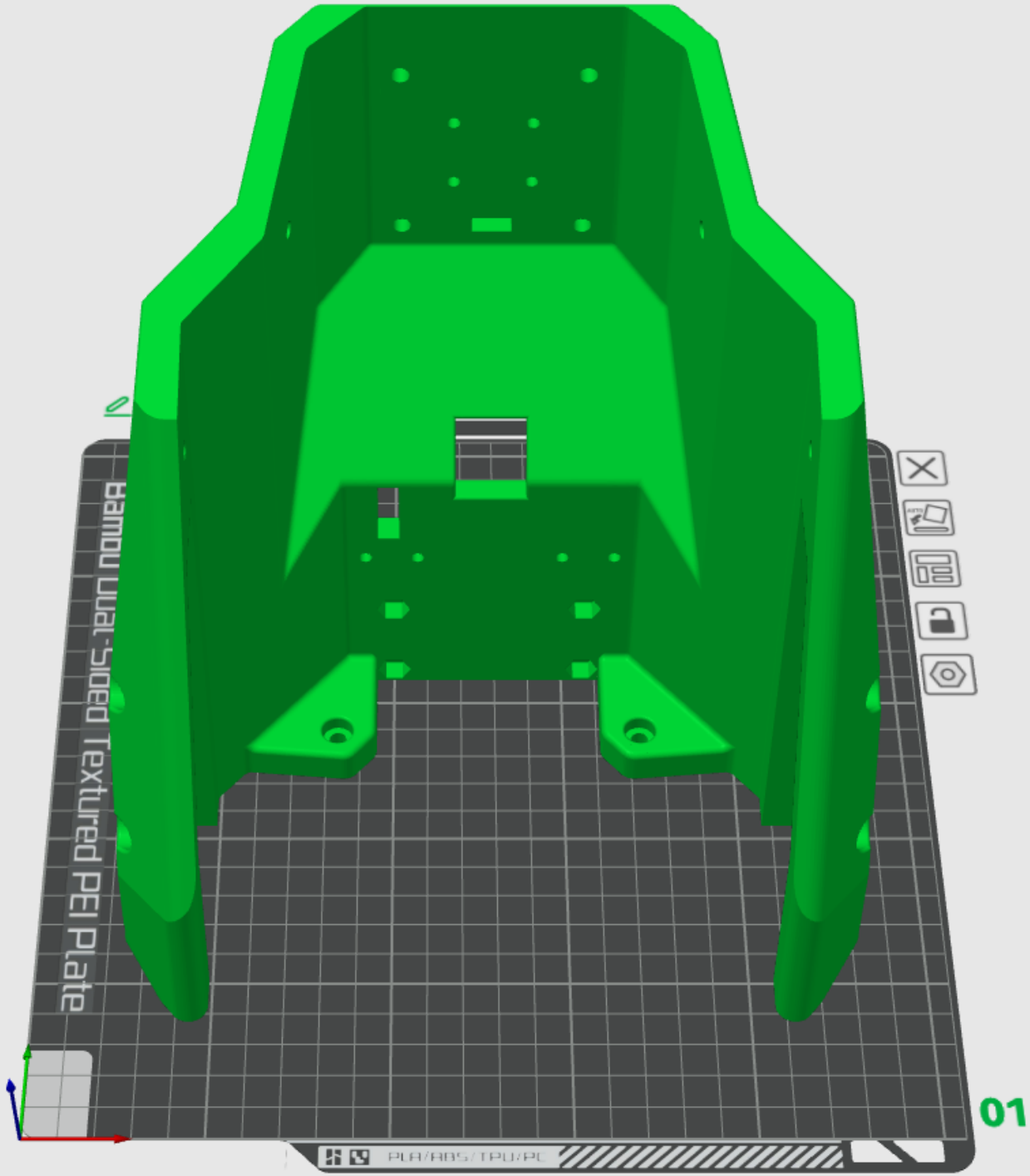


Bottom shell

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

Note:
Support is required

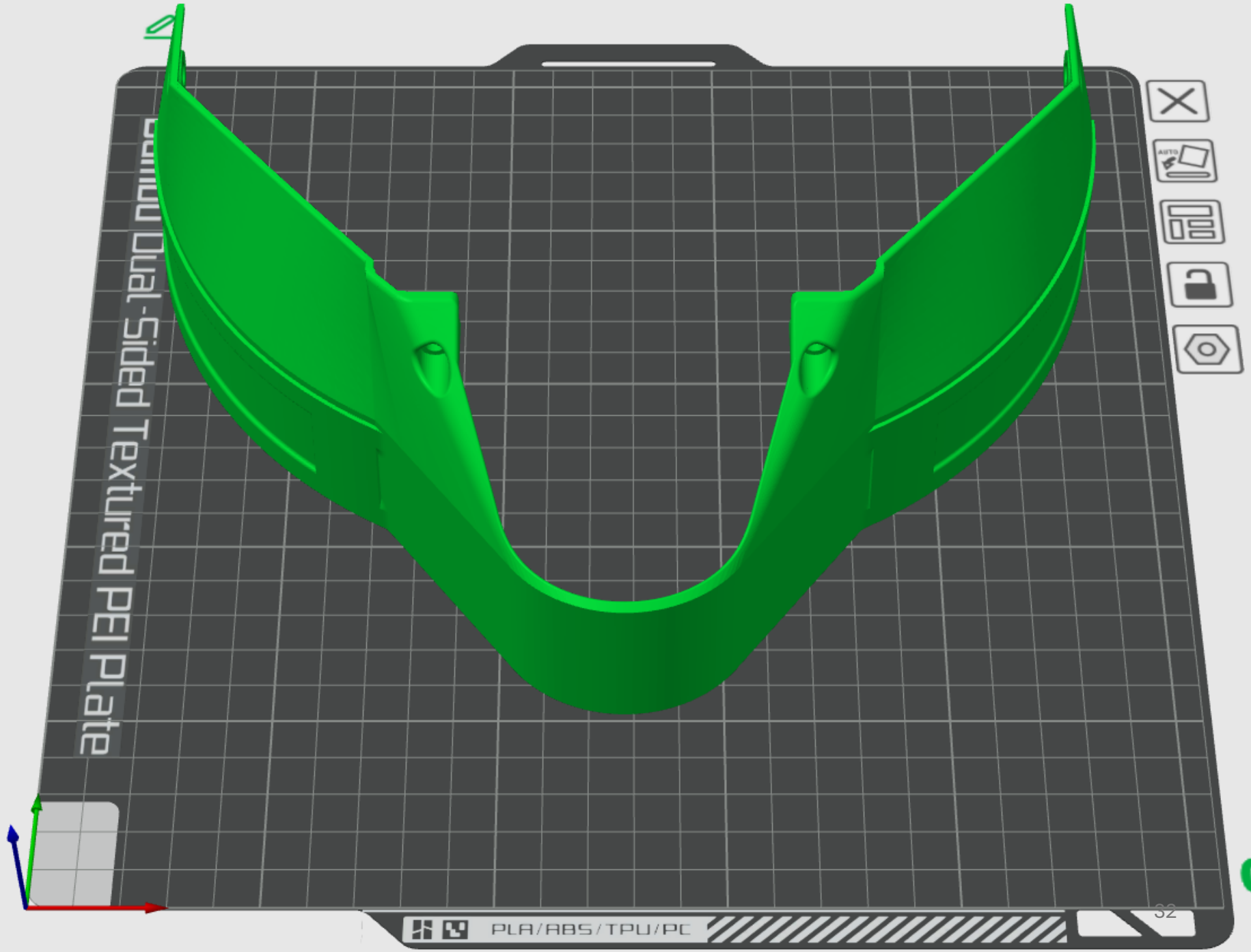


Rear cover

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

Note:
Support is required



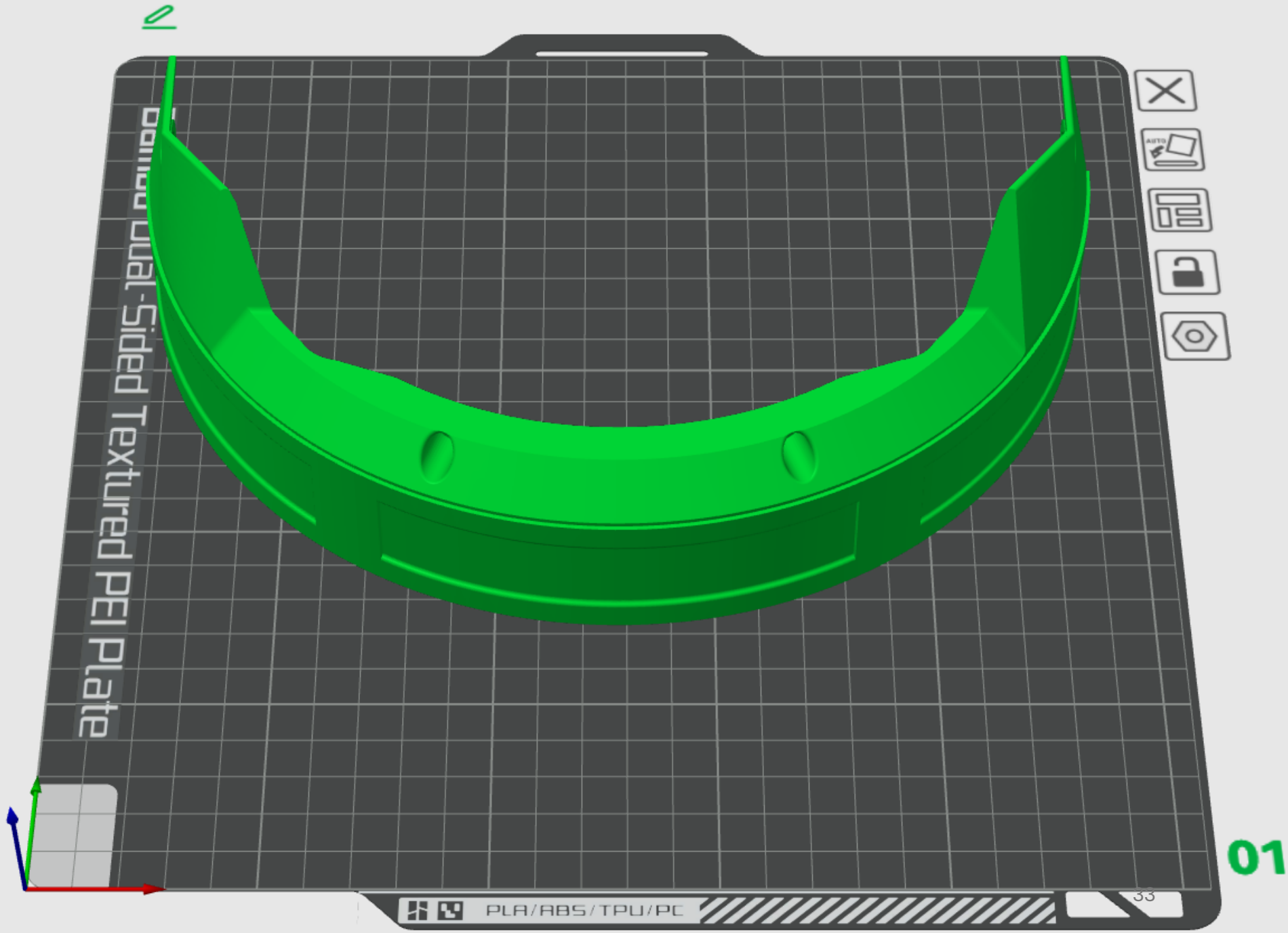
Front cover

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

Note:

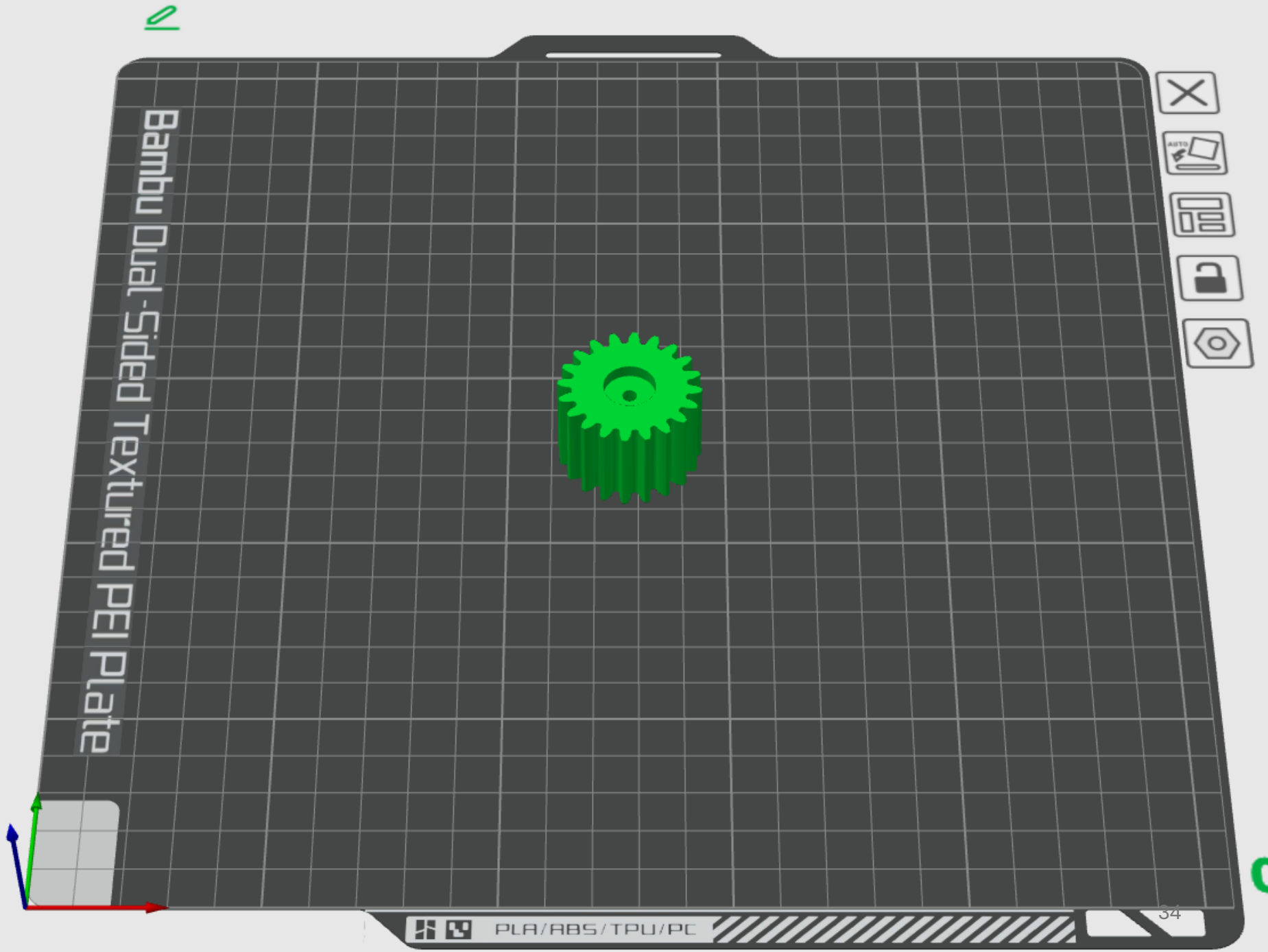
Support is required



Spur gear 20T

Print parameters:

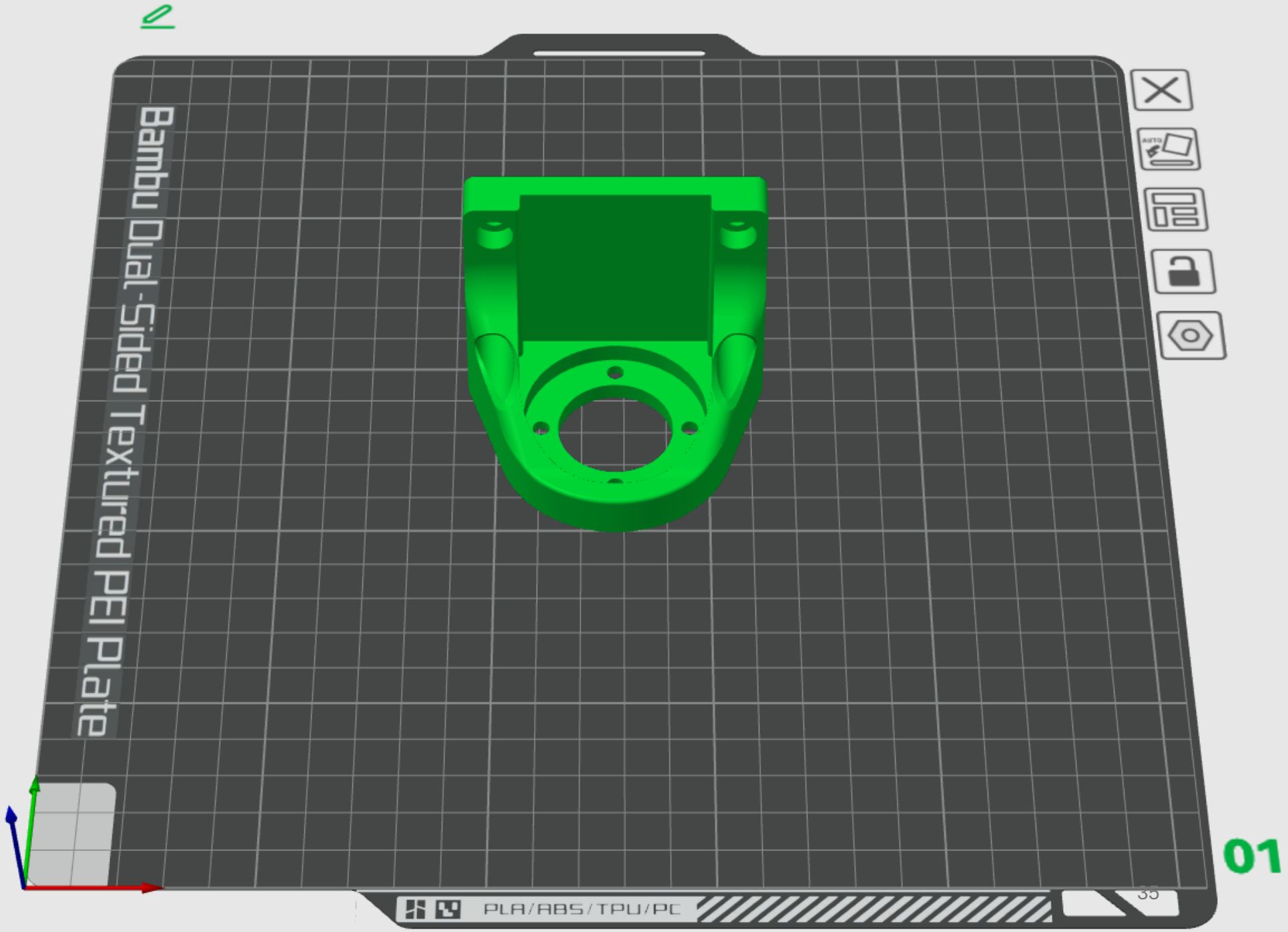
- **Material: PETG**
- **Layer height: 0.1mm**
- **Infill: 100%**
- **Wall loops: 4**



M1 mount

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

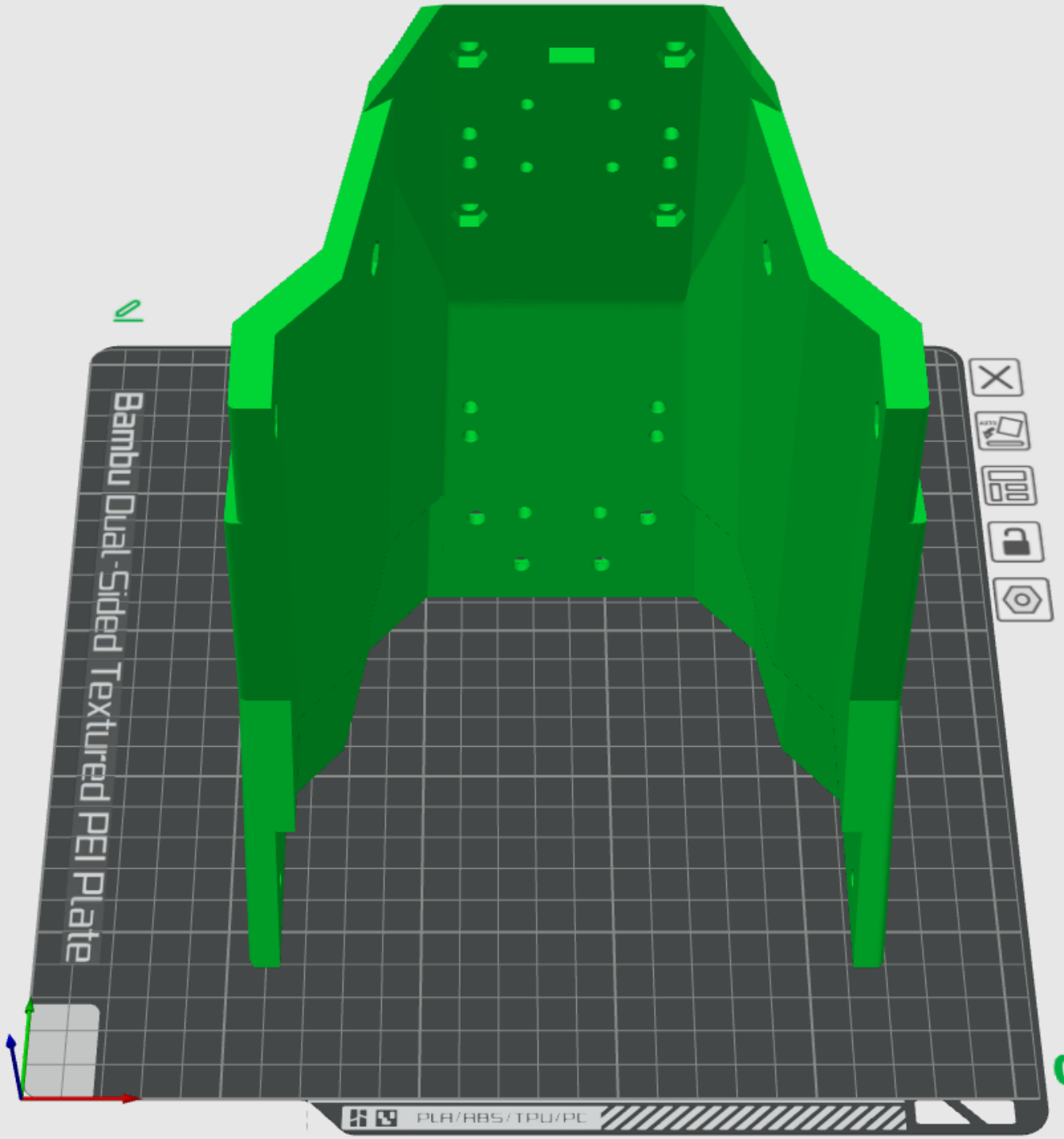


Middle shell

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

Note:
Support is required

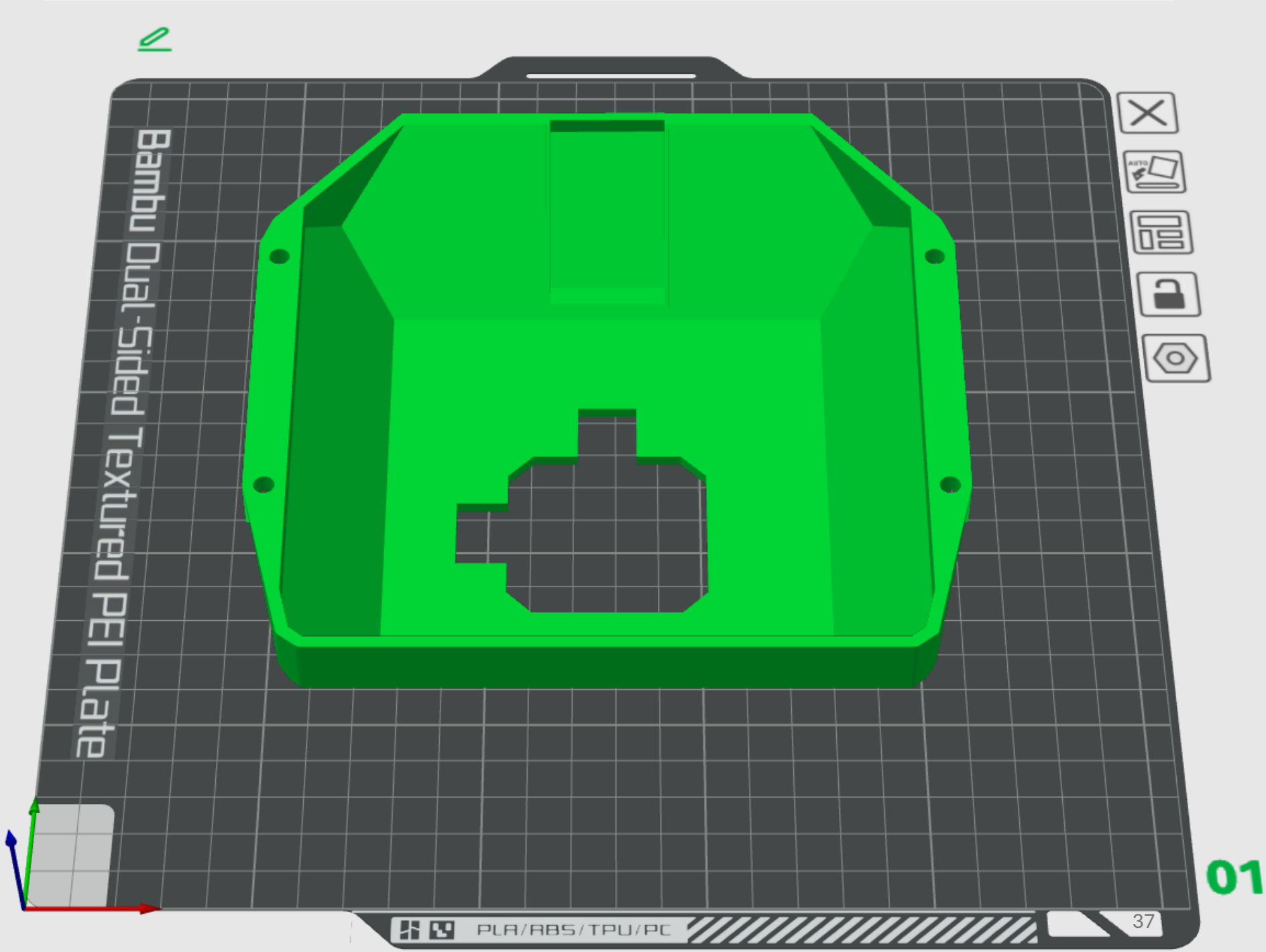


Top cover

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

Note:
Support is required



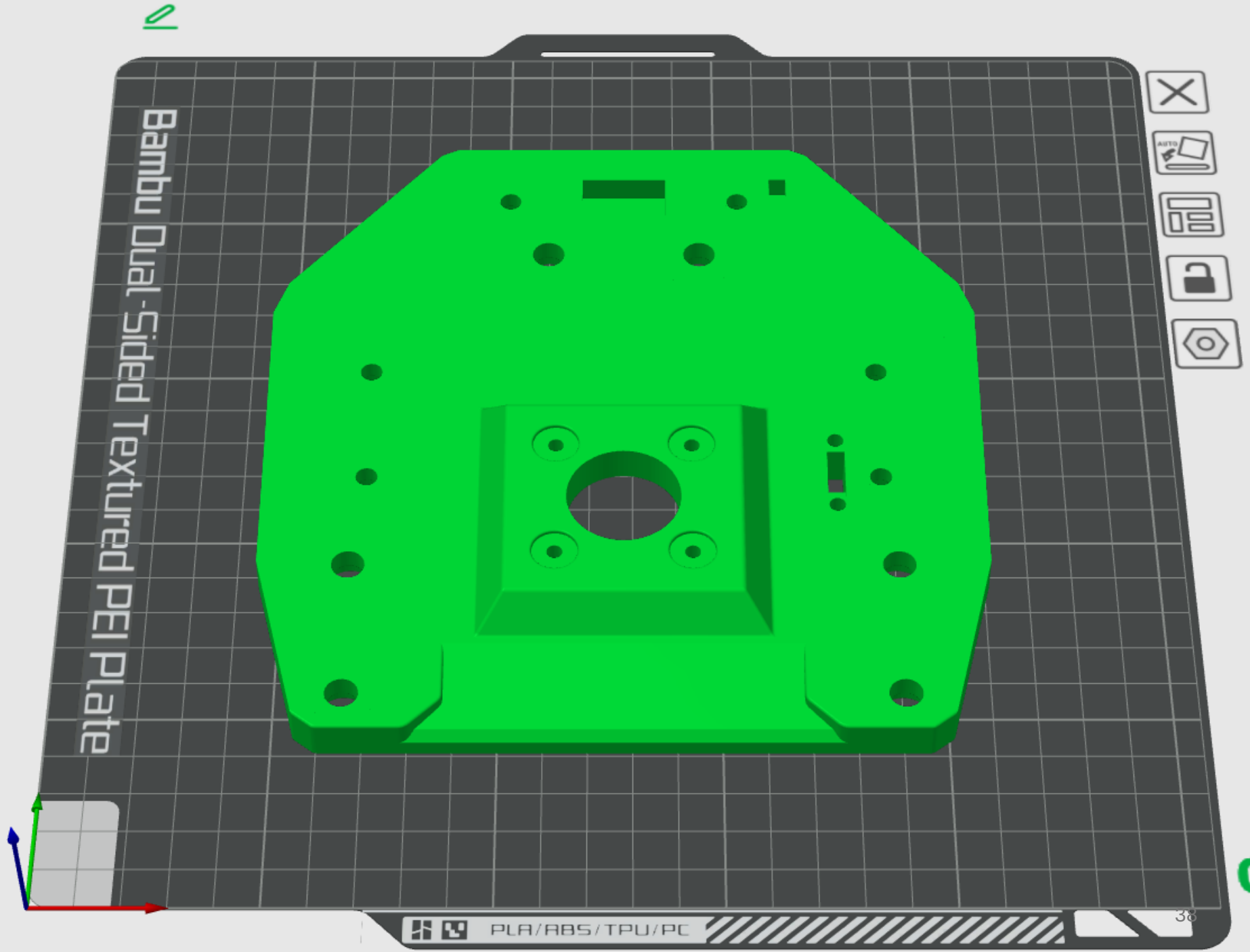
01

Top mount

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

Note:
Support is required

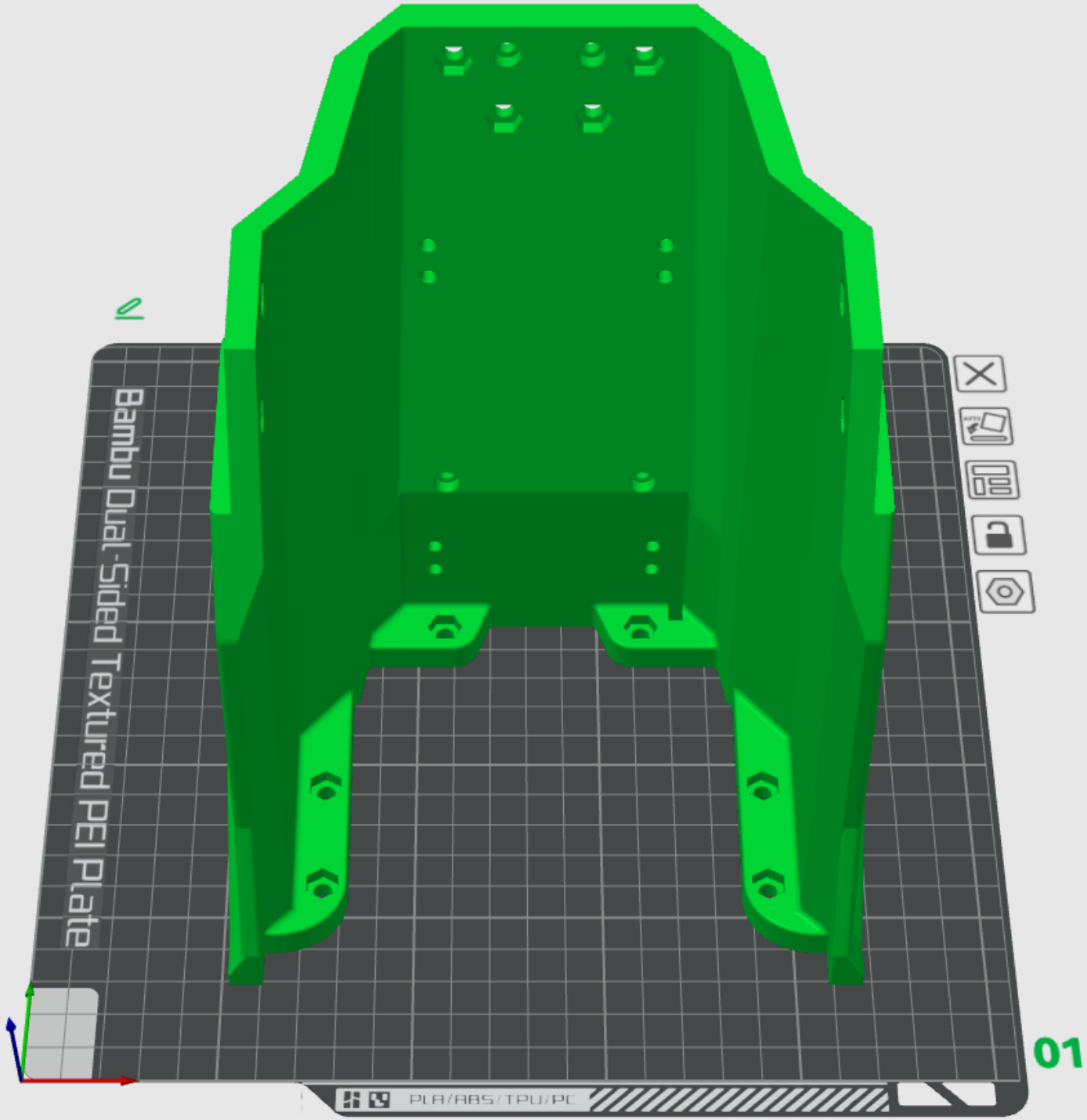


Bottom shell

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

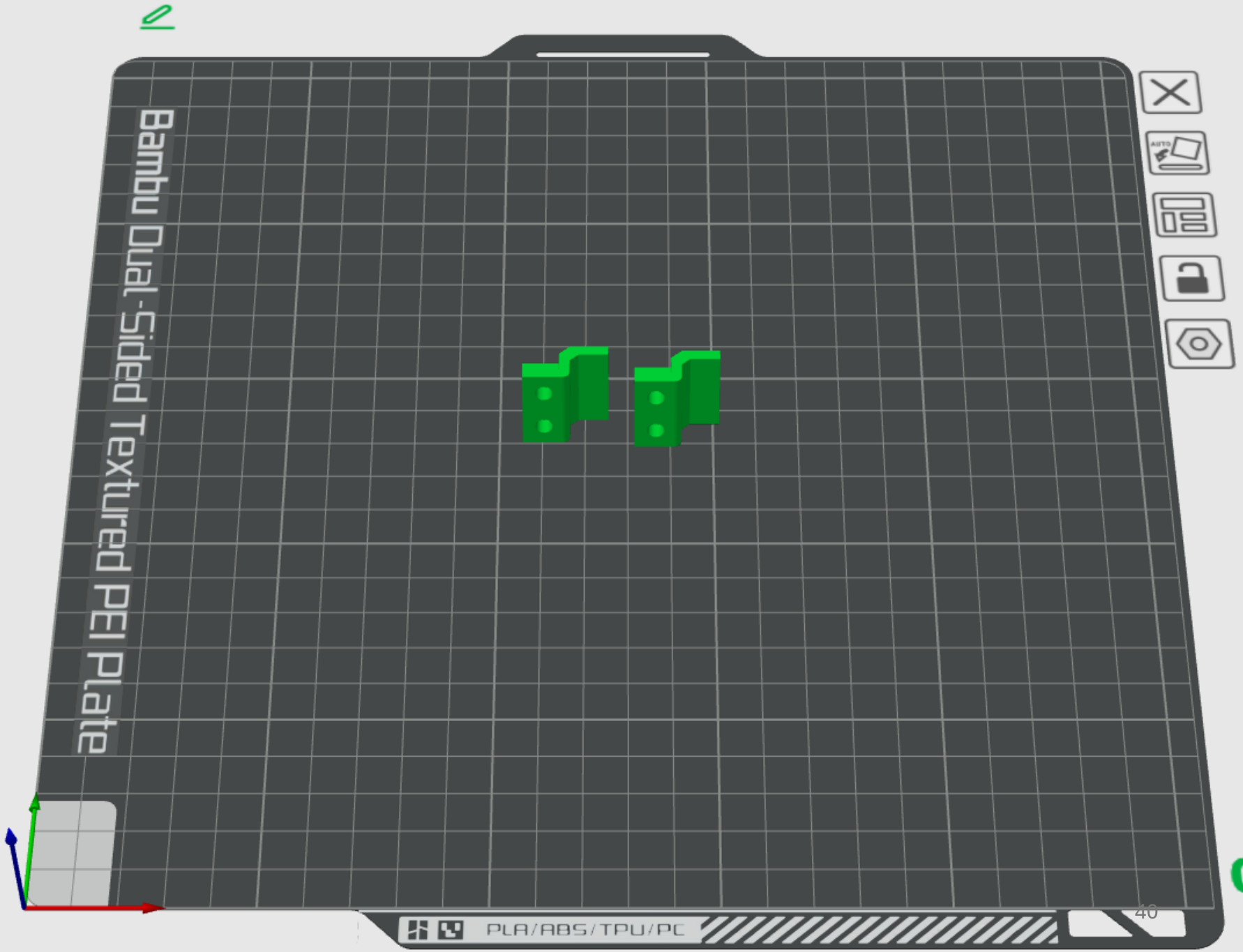
Note:
Support is required



J1 cable holder

Print parameters:

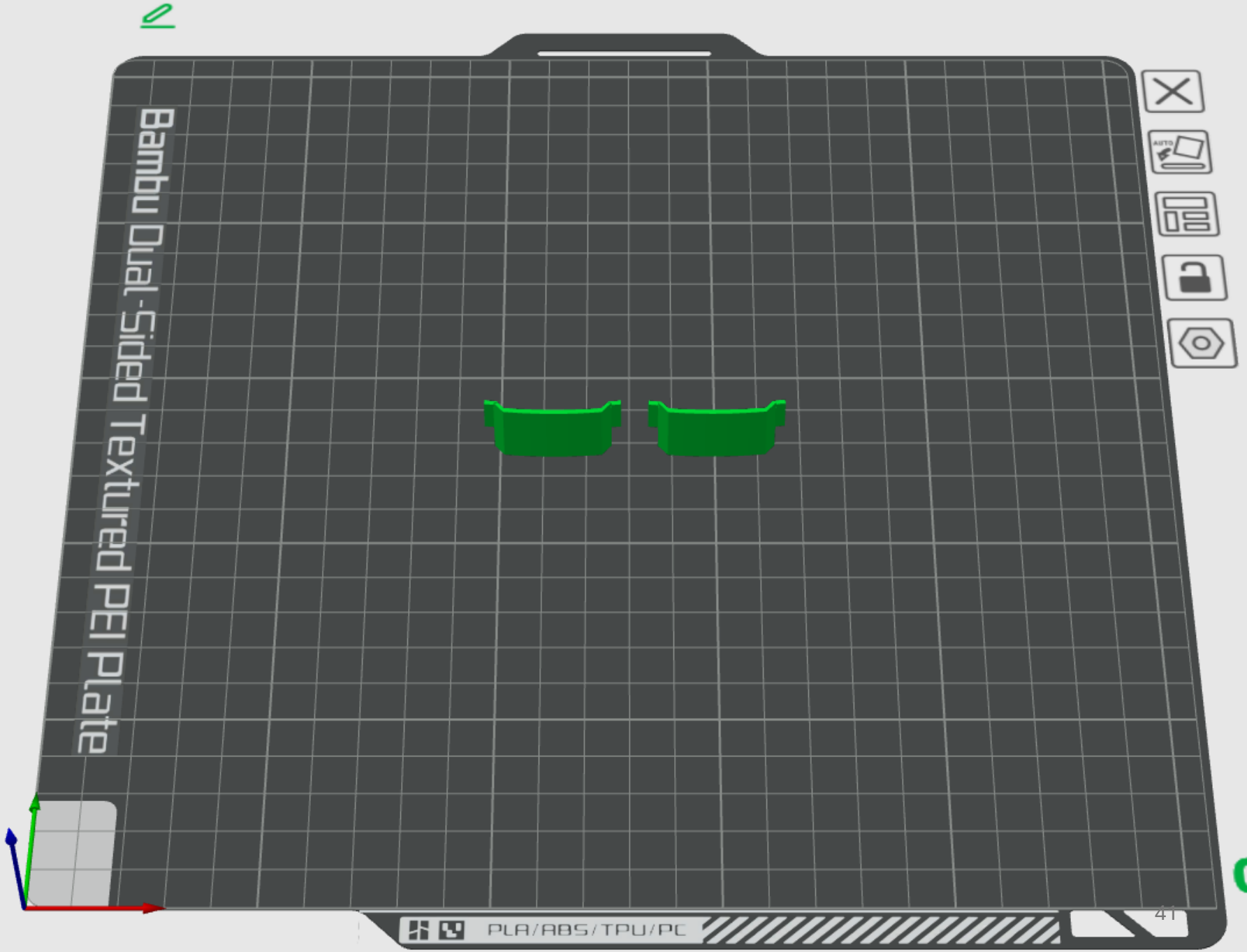
- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**



Cover clip 1&2

Print parameters:

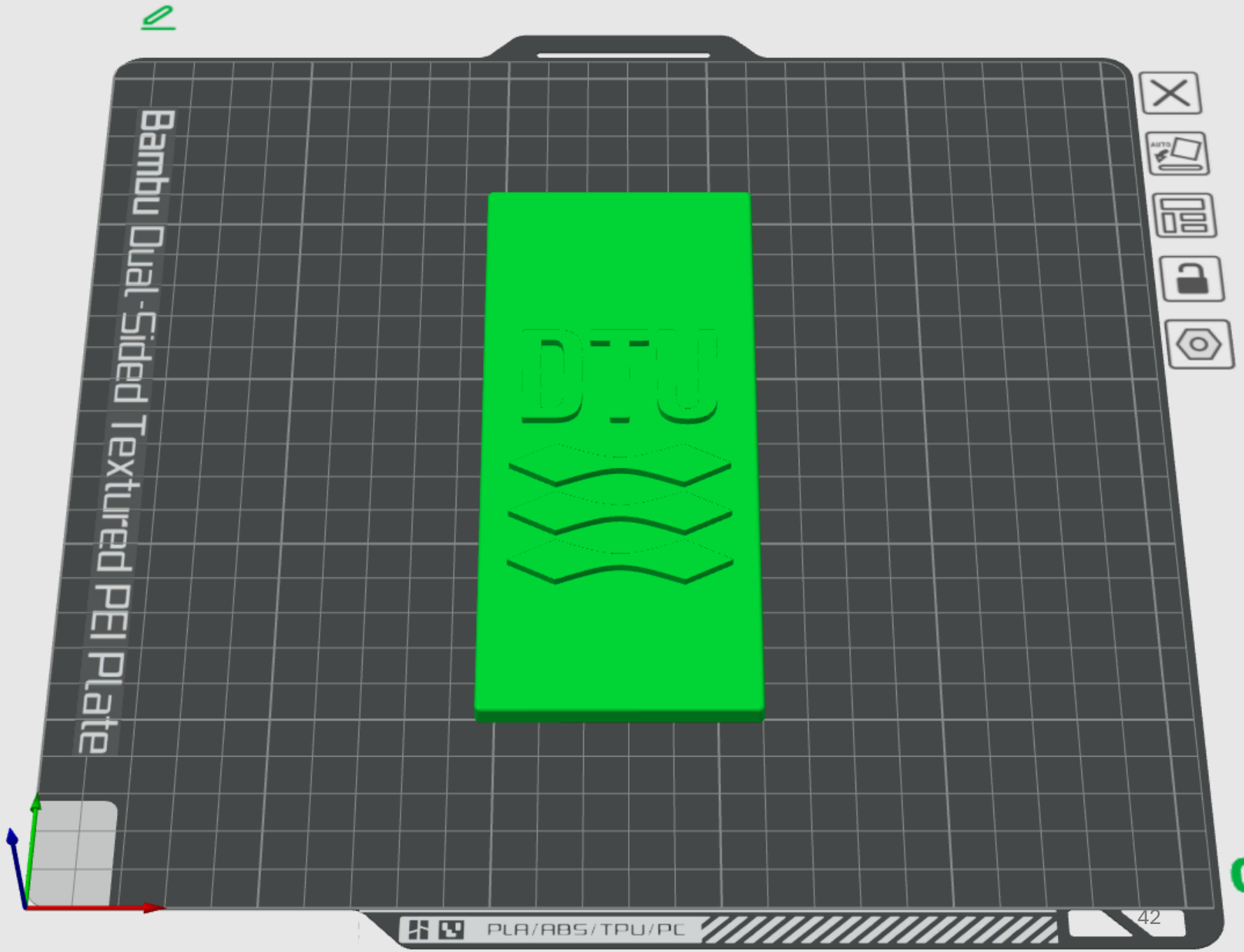
- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**



Logo

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

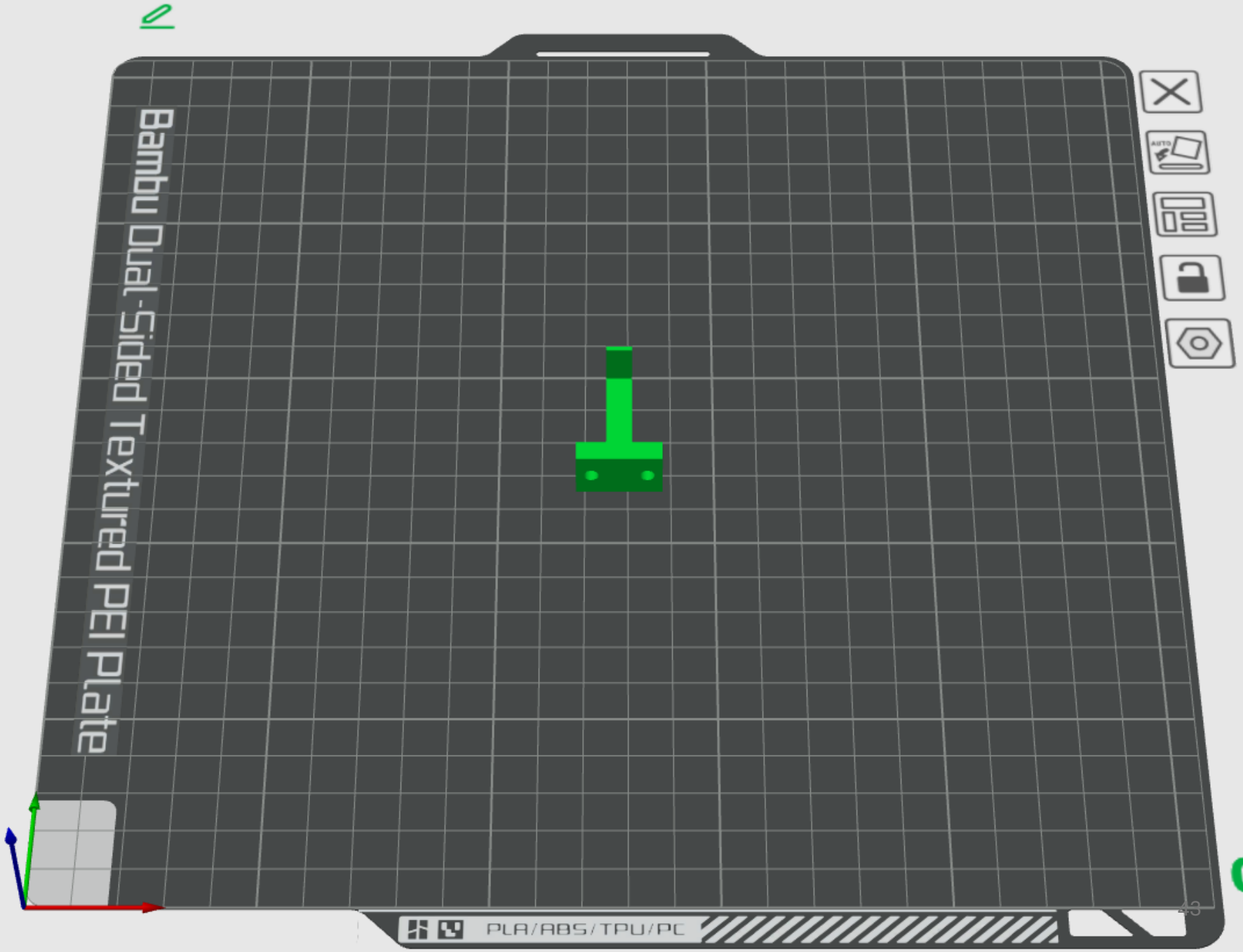


01

J1 sensor trigger

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**



01

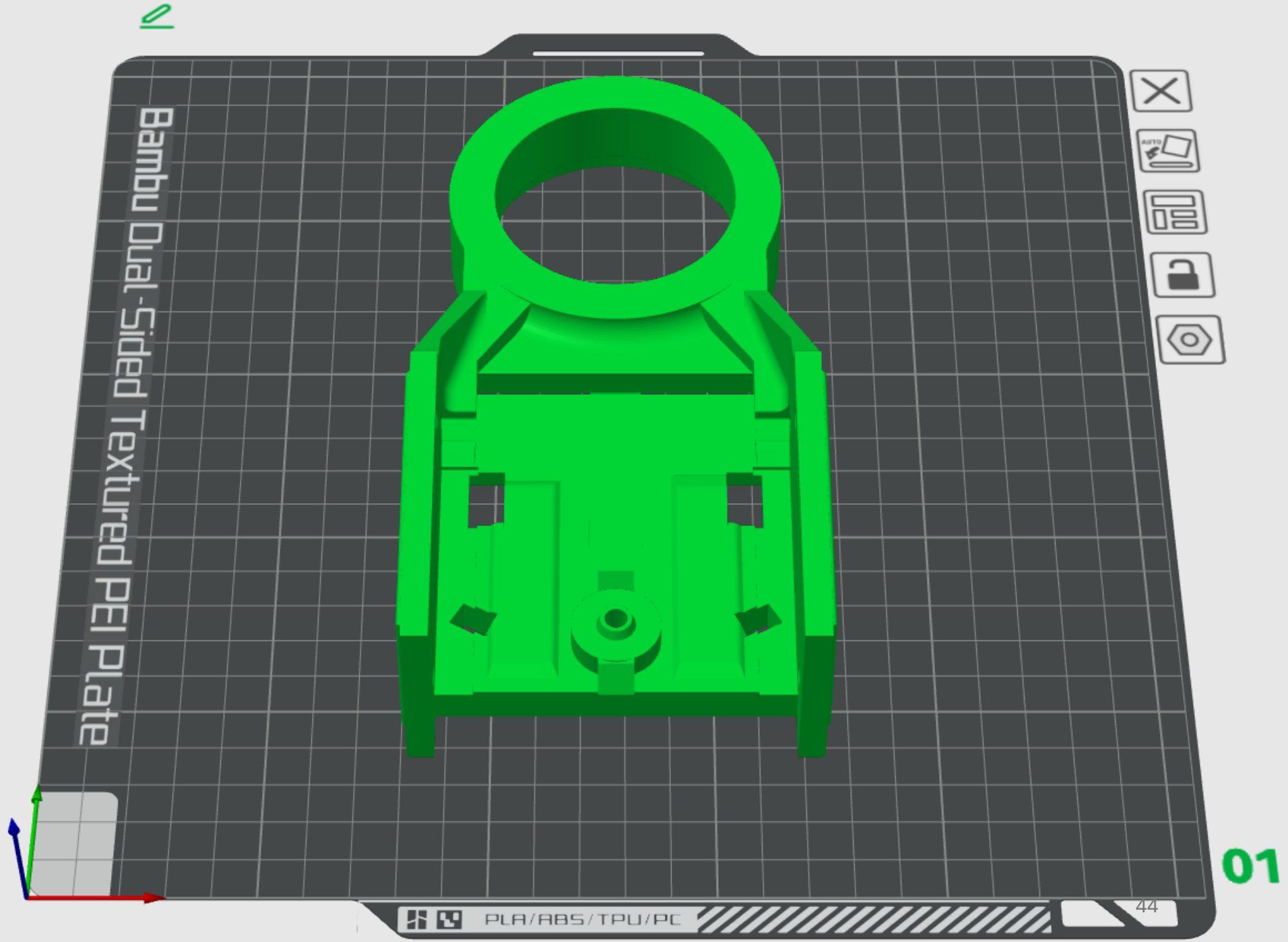
Arm link top

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

Note:

Support is required

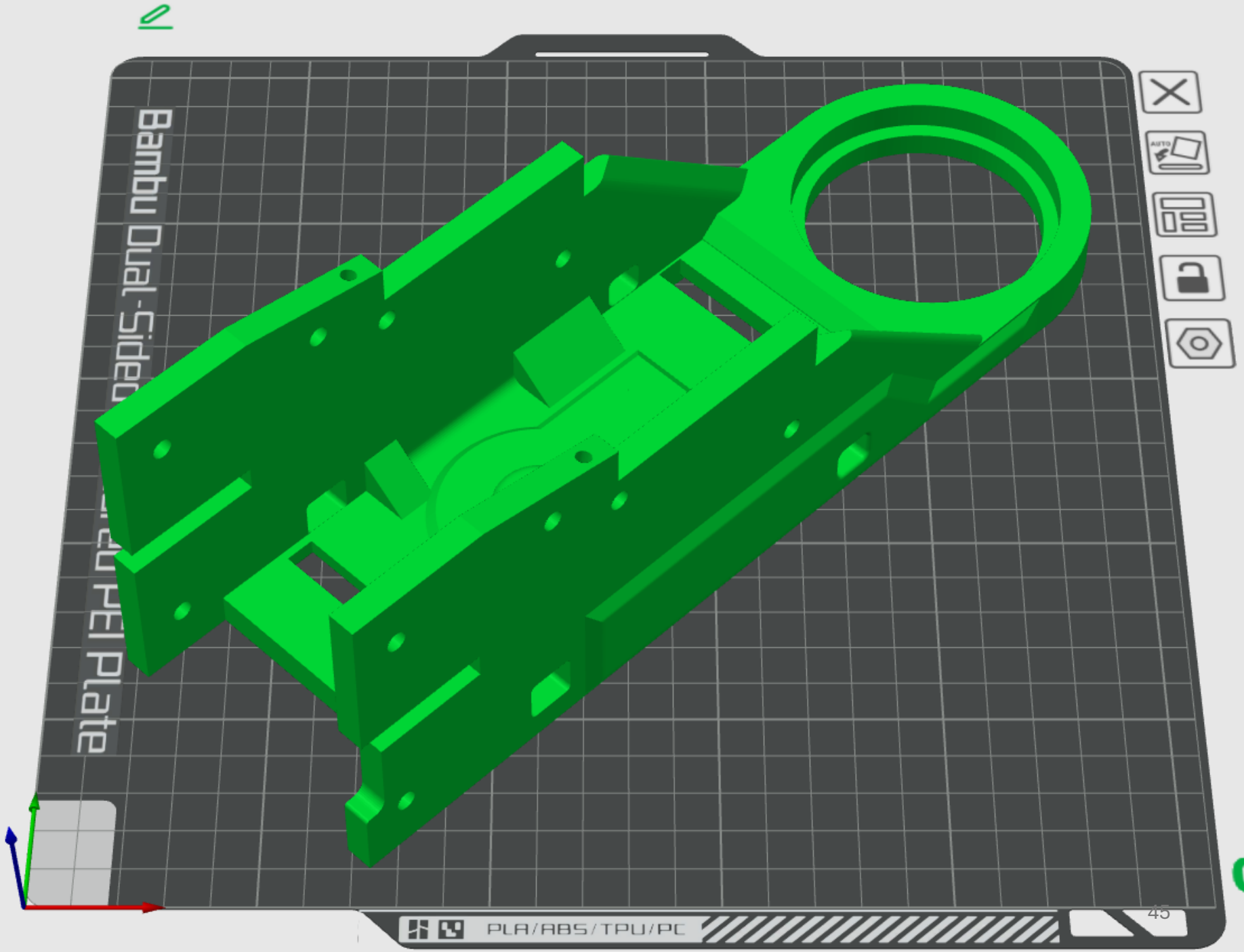


Arm link bottom

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

Note:
Support is required

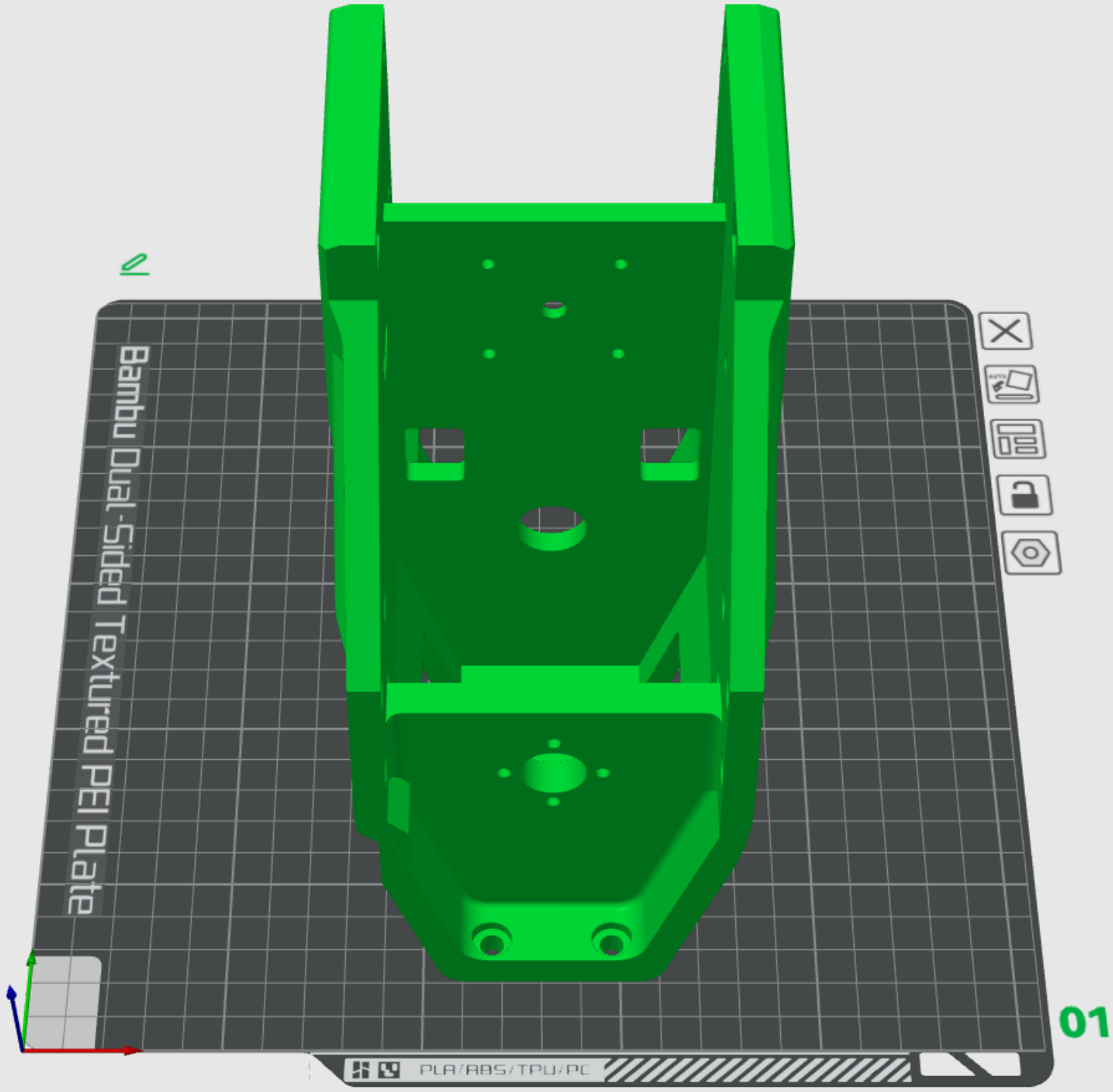


Arm mount

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

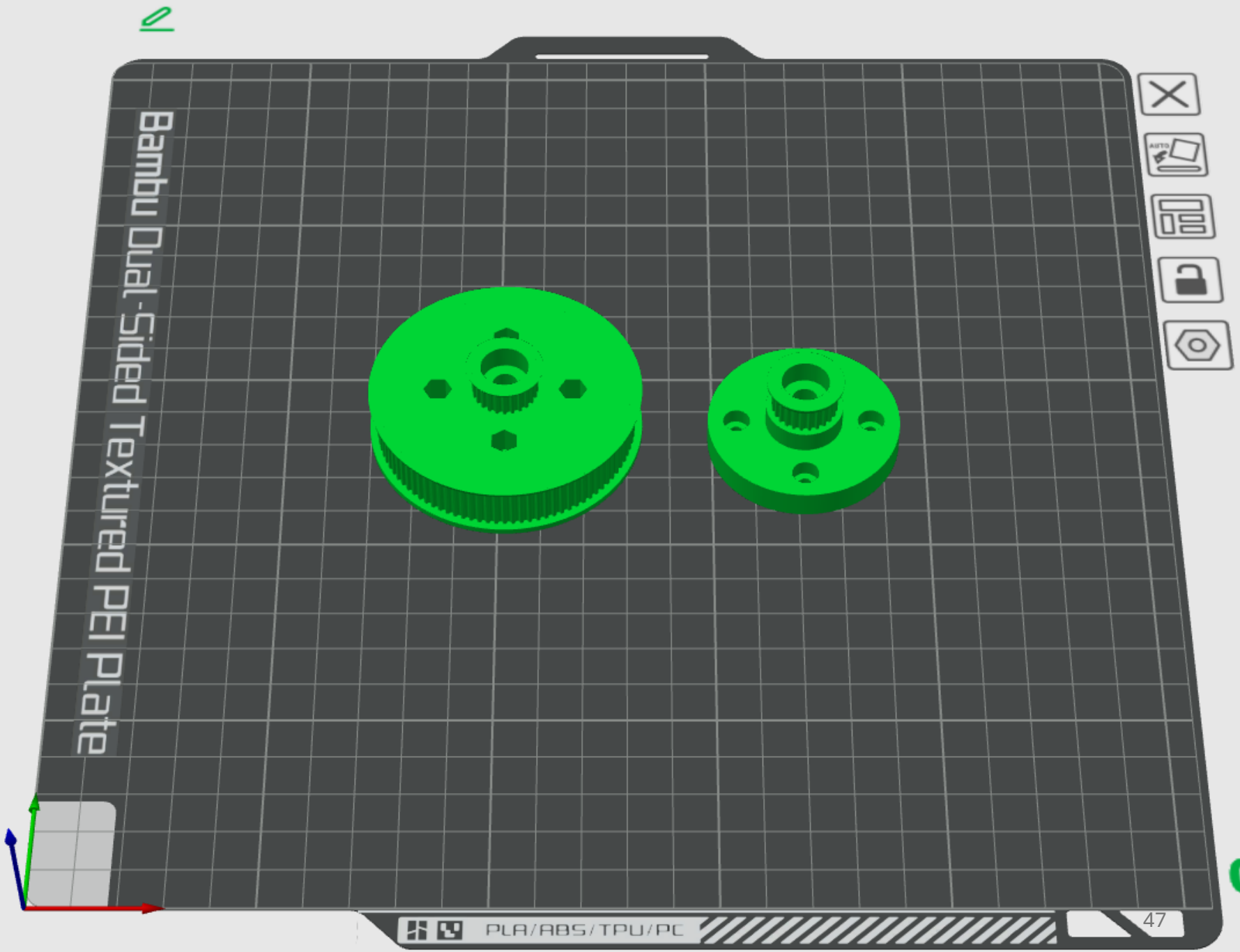
Note:
Support is required



Gear 25T & 96T

Print parameters:

- **Material: PETG**
- **Layer height: 0.1mm**
- **Infill: 100%**
- **Wall loops: 4**

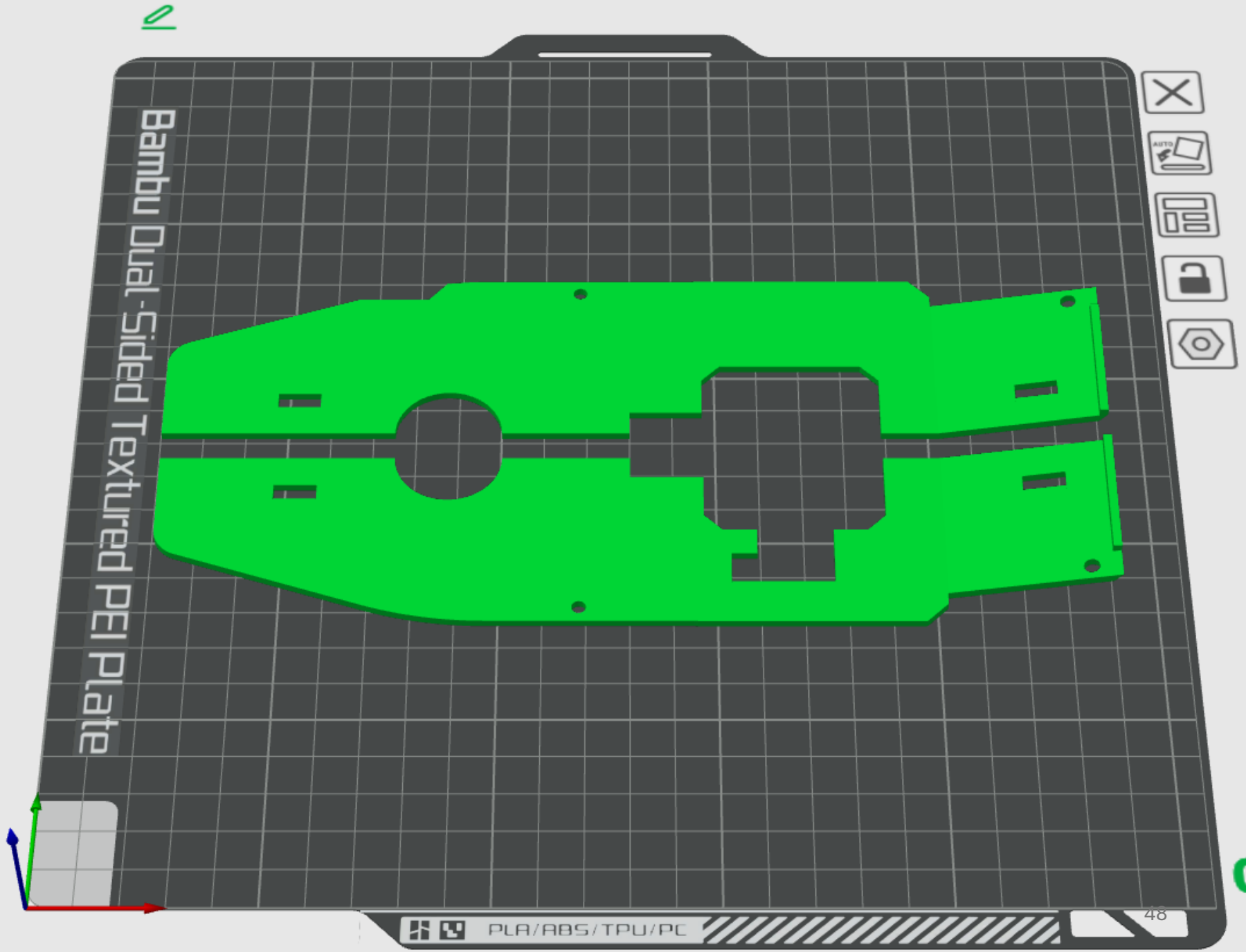


Arm cover 1&2

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

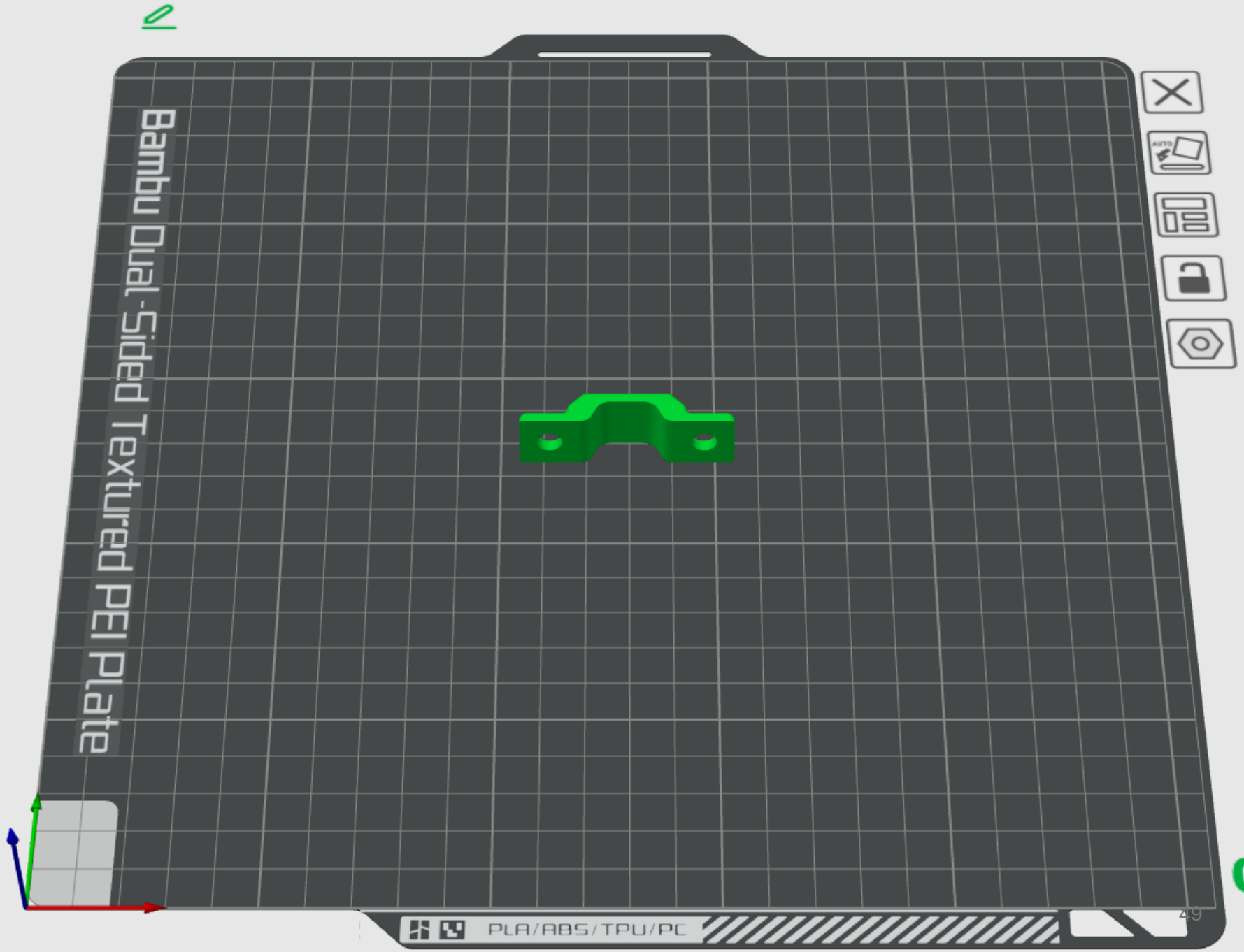
Note:
Support is required



J2 Cable holder

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

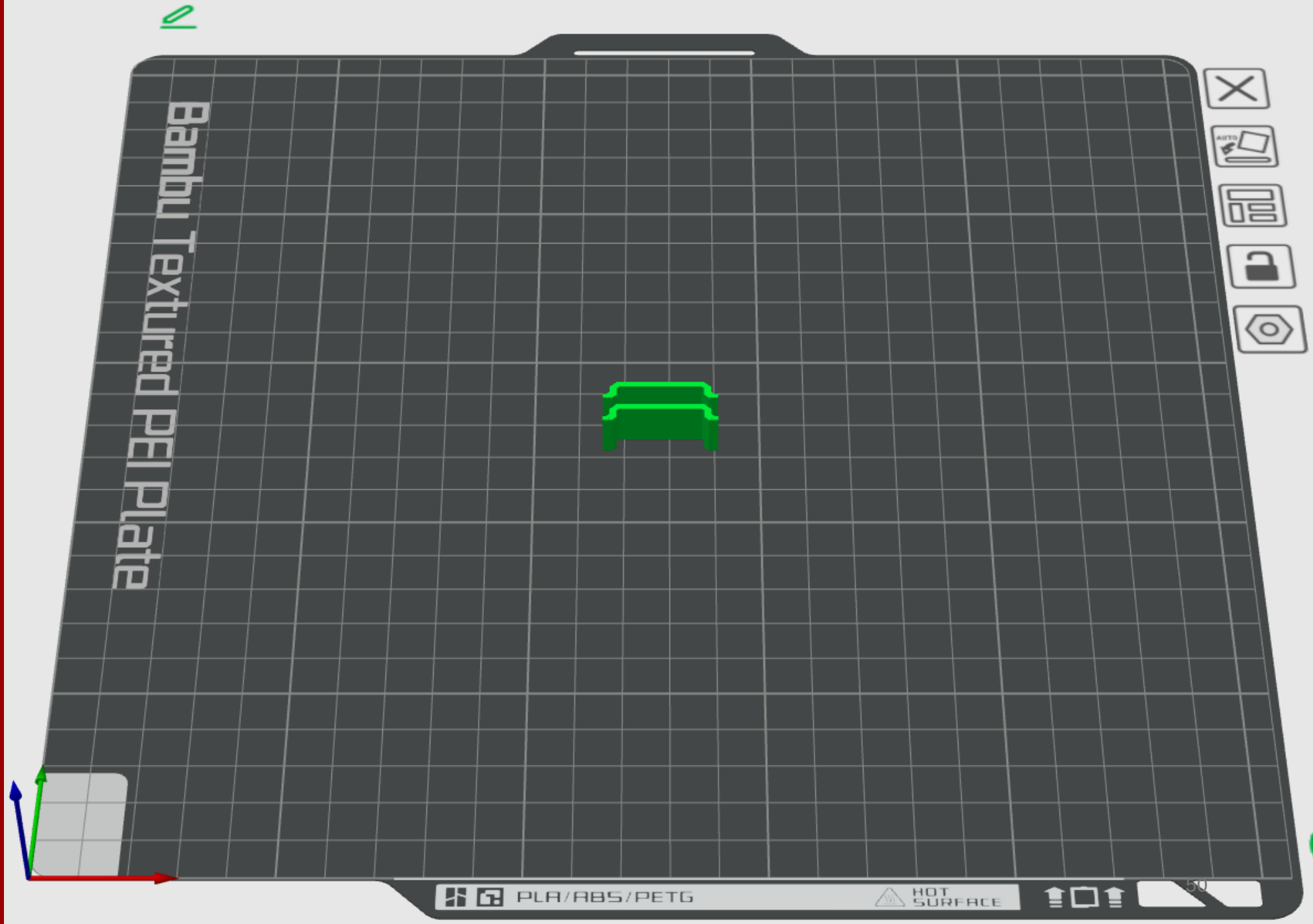


01

J2 Cover clip 1&2

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**



J2 Belt tensioner 2 1&2&3&4

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

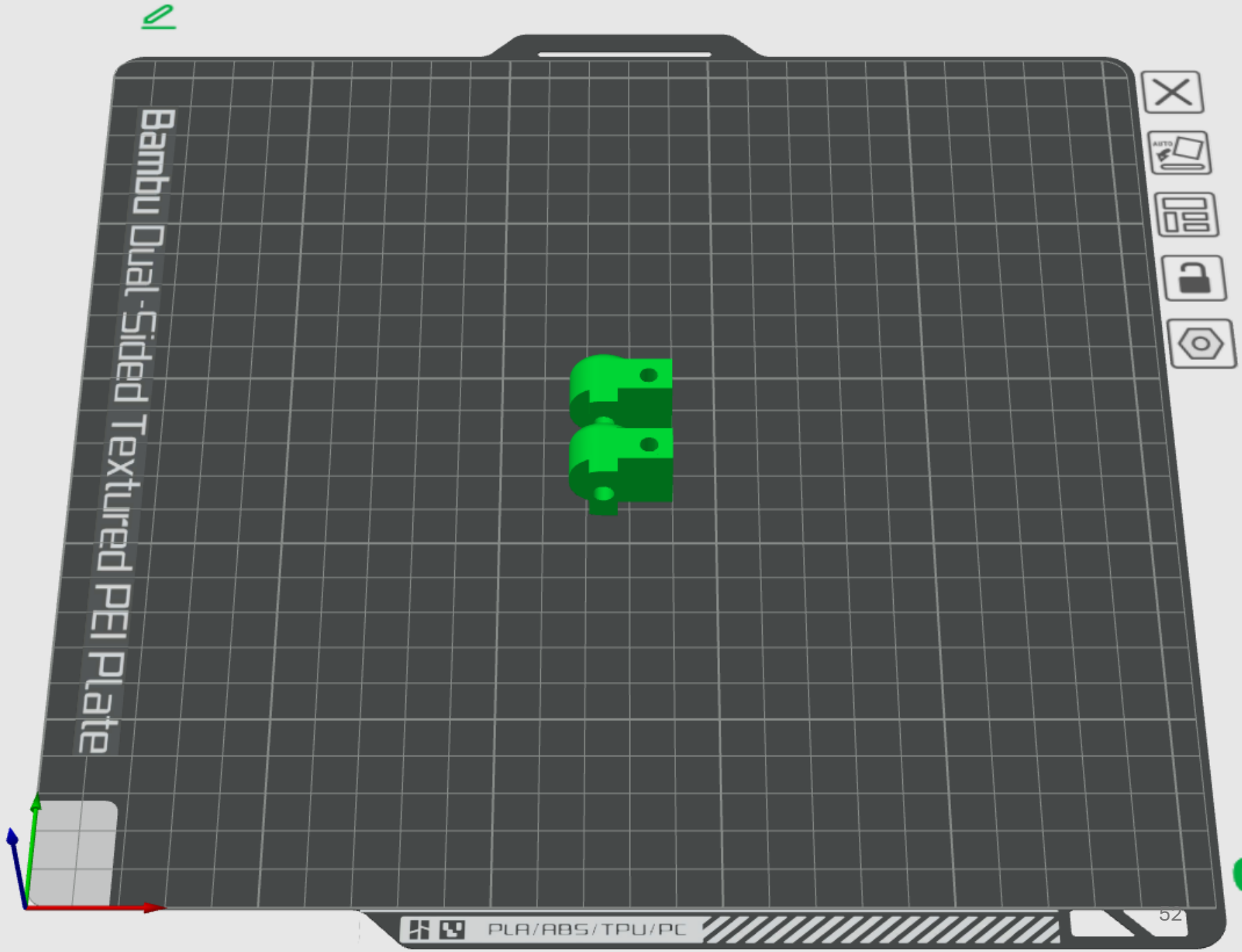
Note:
Support is required

J2 Belt tensioner 1 1&2

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

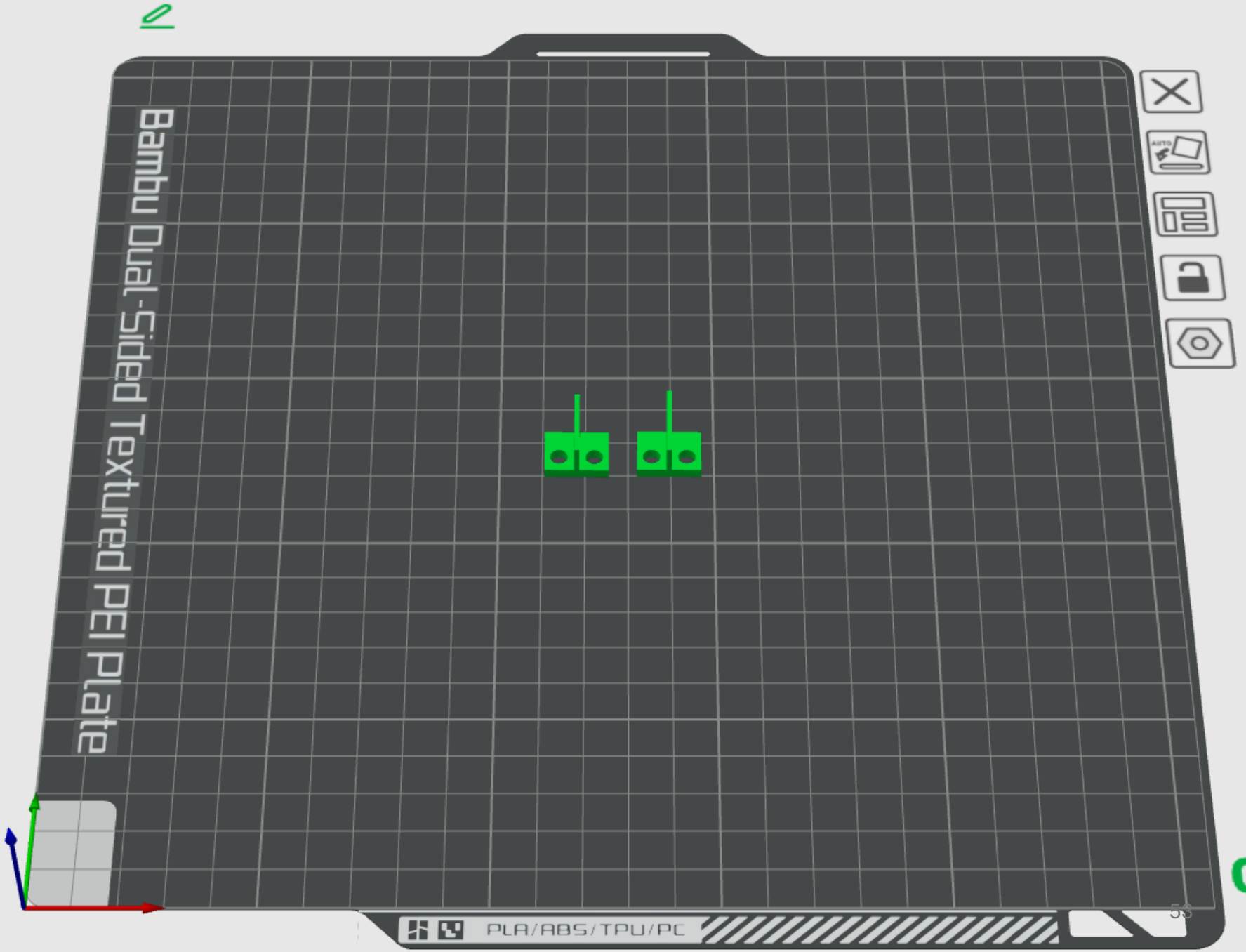
Note:
Support is required



J2 Sensor trigger 1&2

Print parameters:

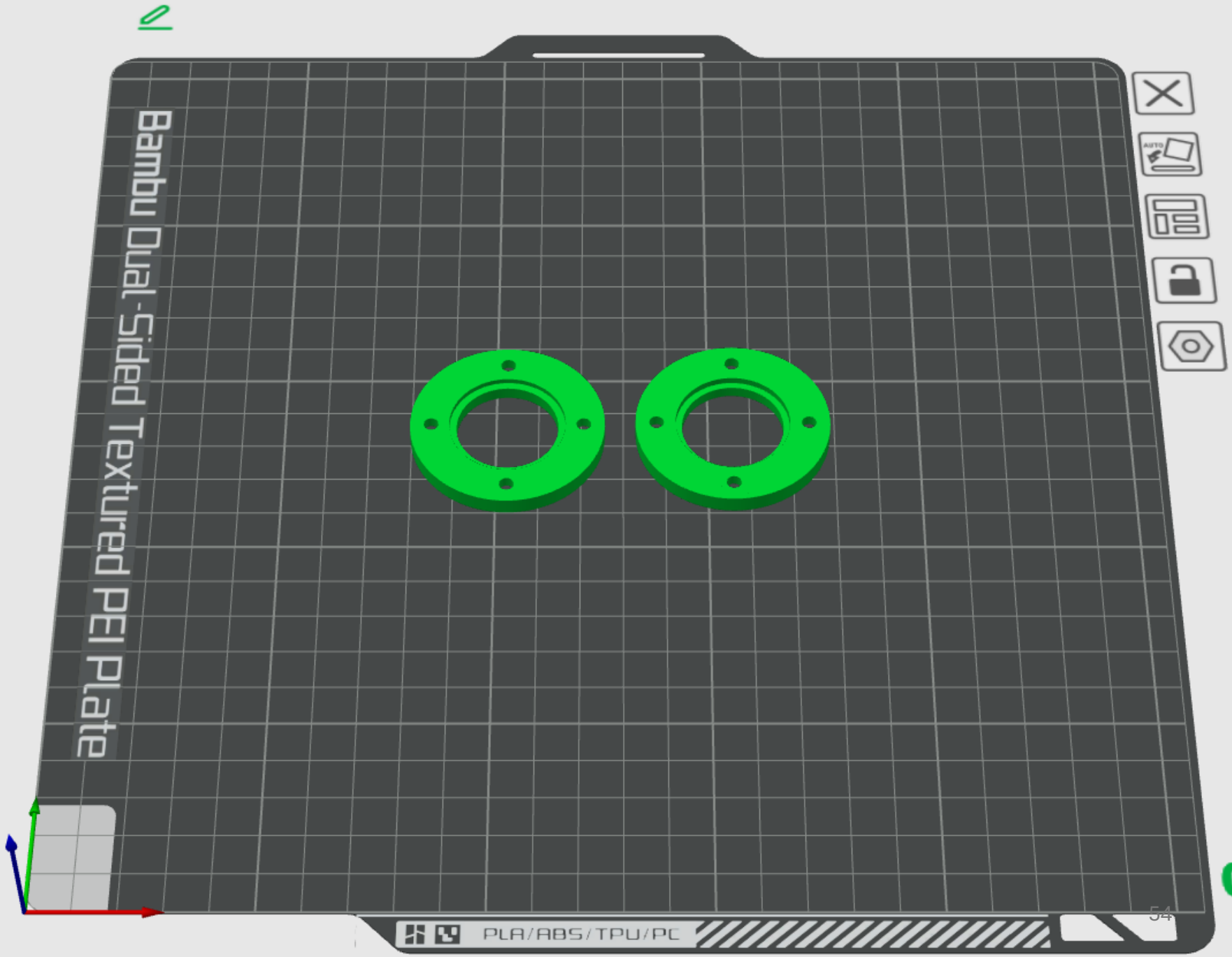
- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**



BB mount bottom&top

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**



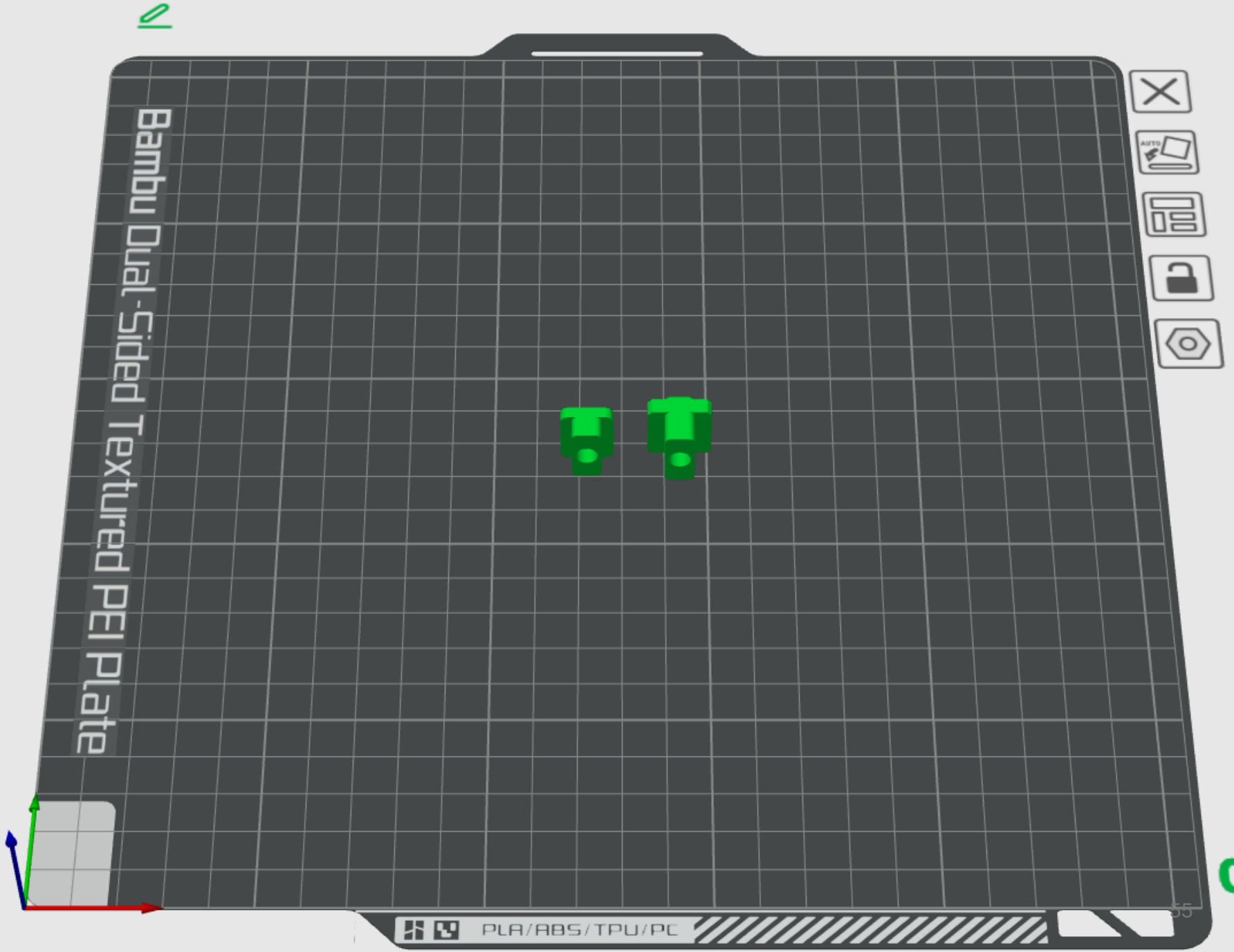
01

J3 Belt tensioner 1&2

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

Note:
Support is required

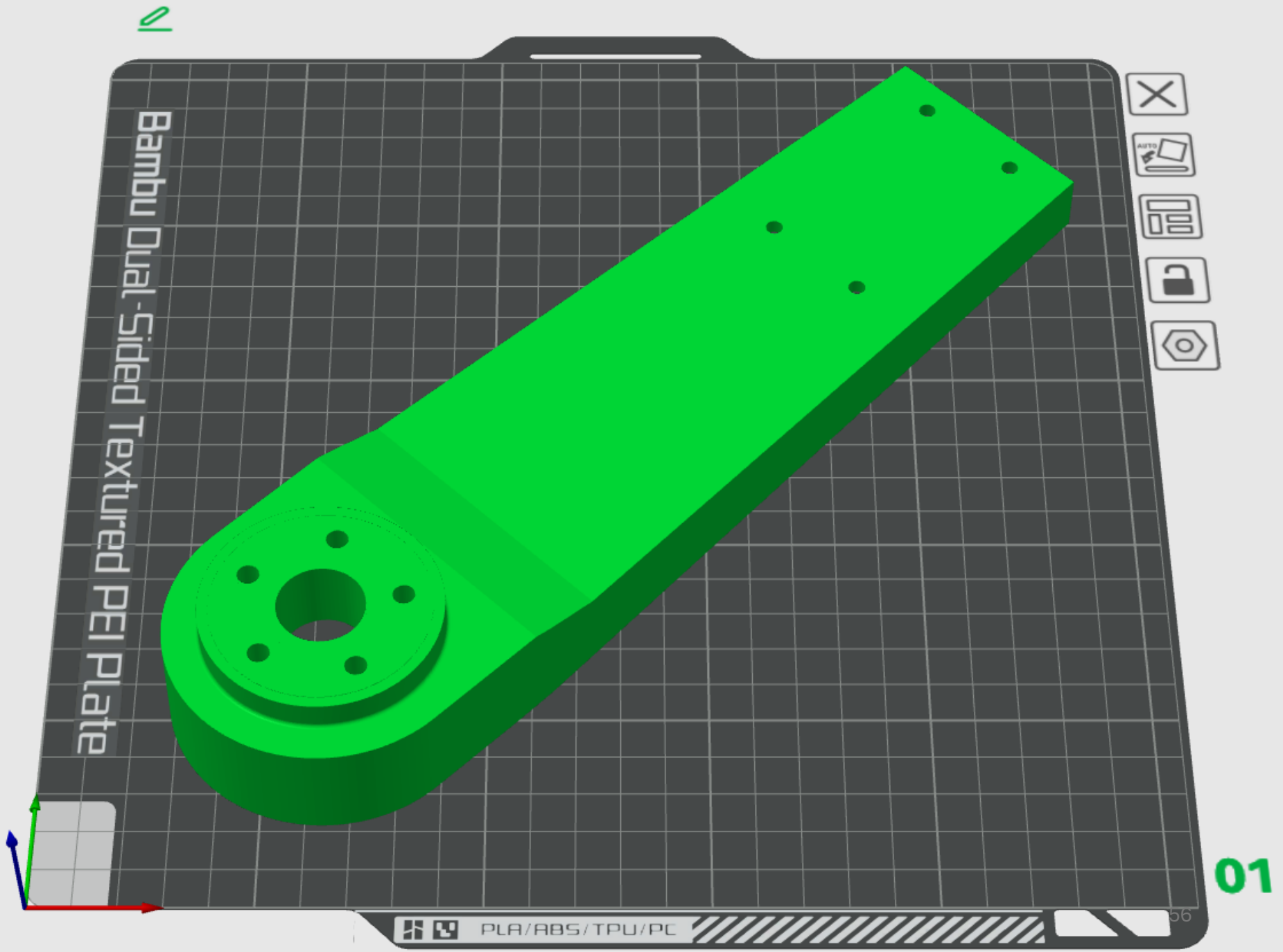


Bottom link long

Print parameters:

- **Material: PLA**
- **Layer height: 0.2mm**
- **Infill: 35%**
- **Wall loops: 5**

Note:
Support is required

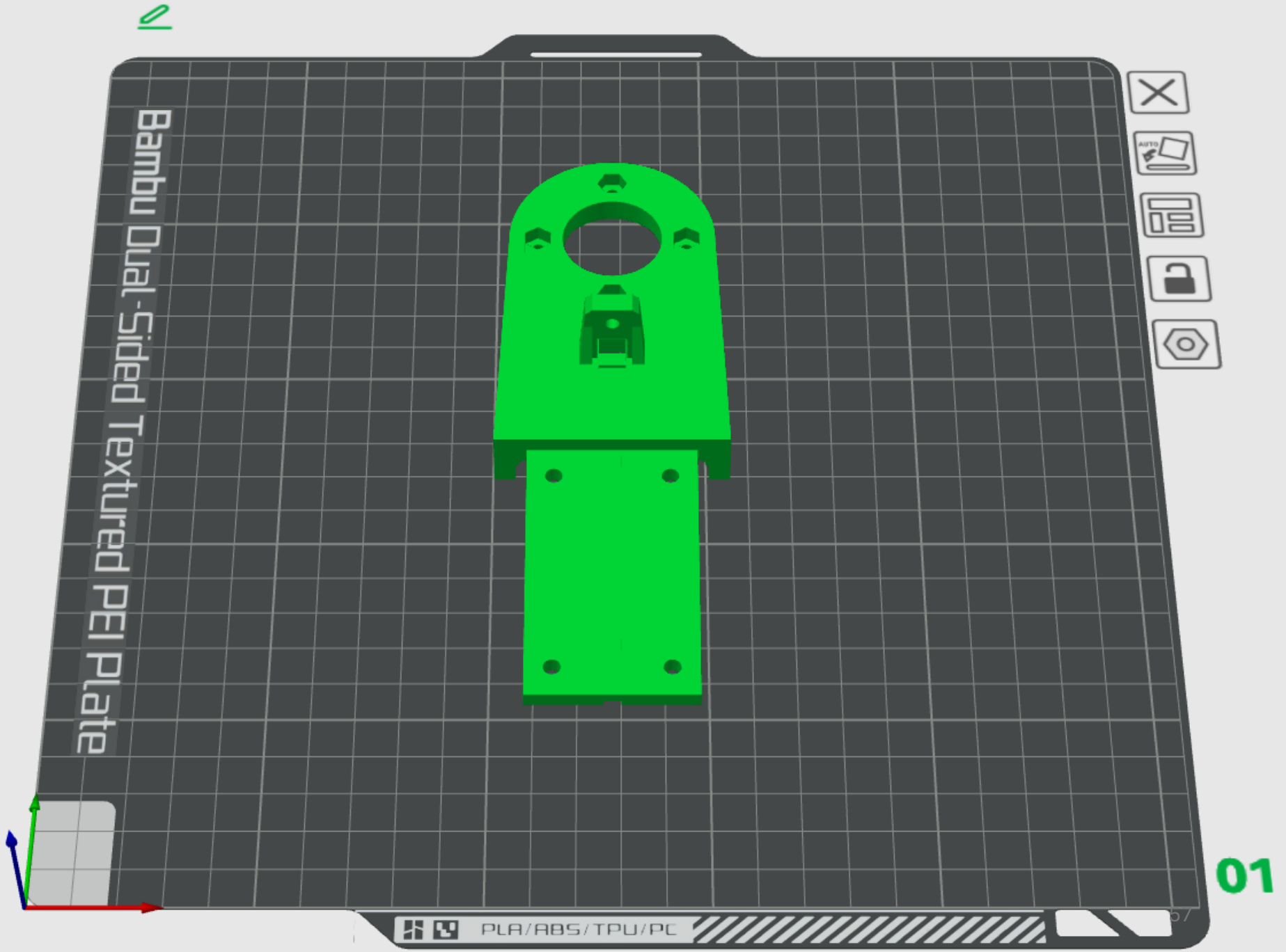


Bottom link short

Print parameters:

- **Material: PLA**
- **Layer height: 0.2mm**
- **Infill: 35%**
- **Wall loops: 5**

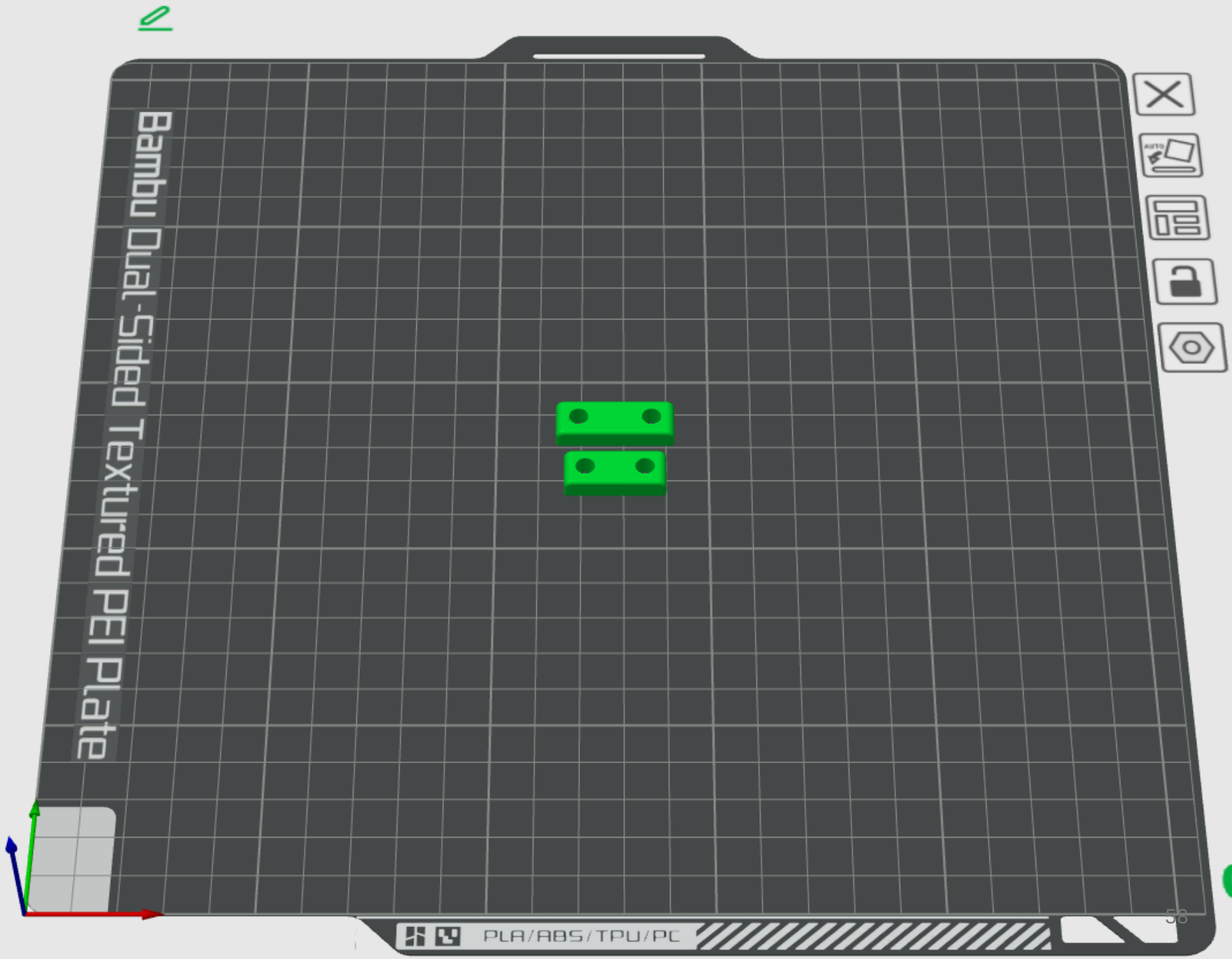
Note:
Support is required



J3 Cable holder 1&2

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

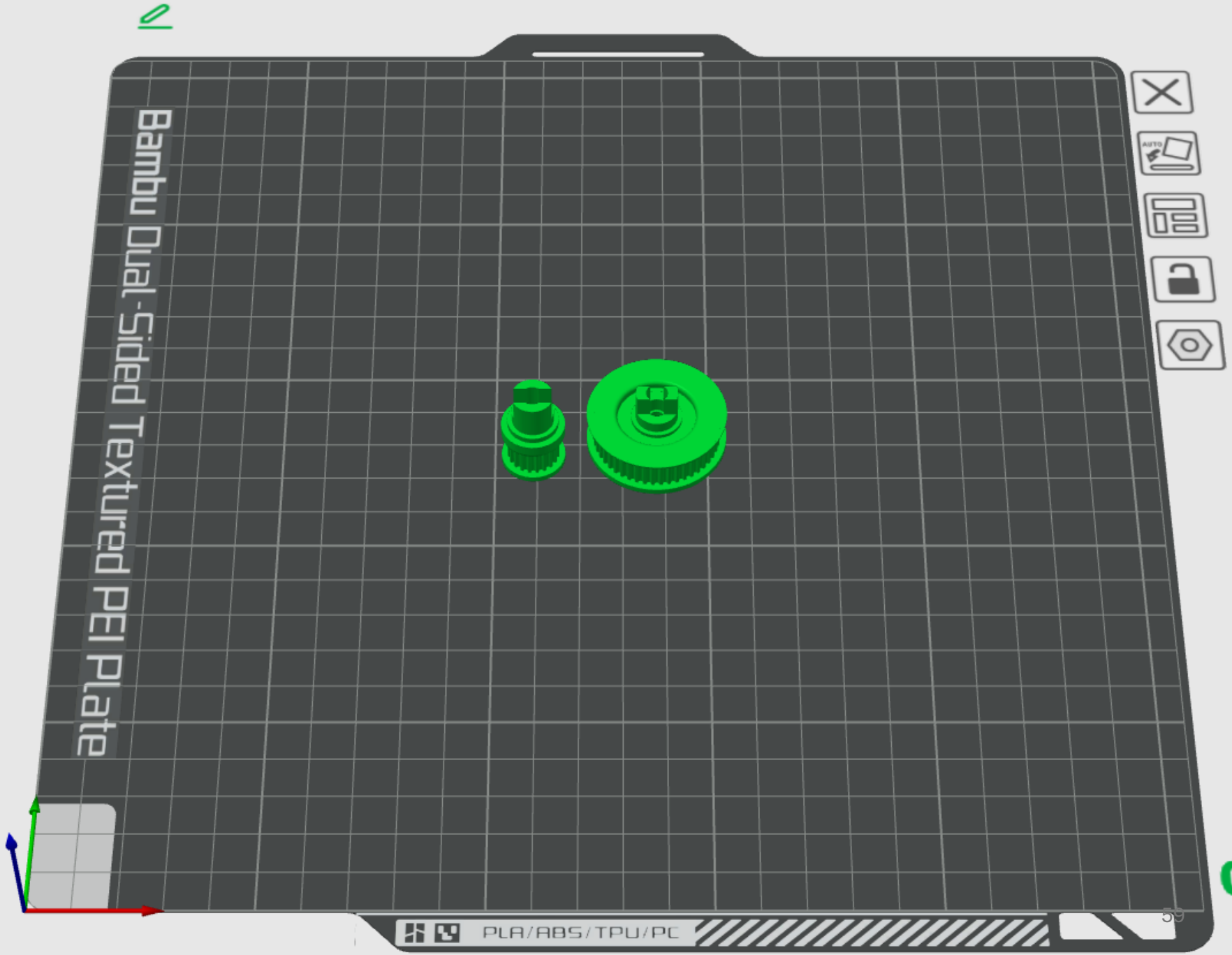


01

Gear 20T & 48T

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 100%**
- **Wall loops: 4**

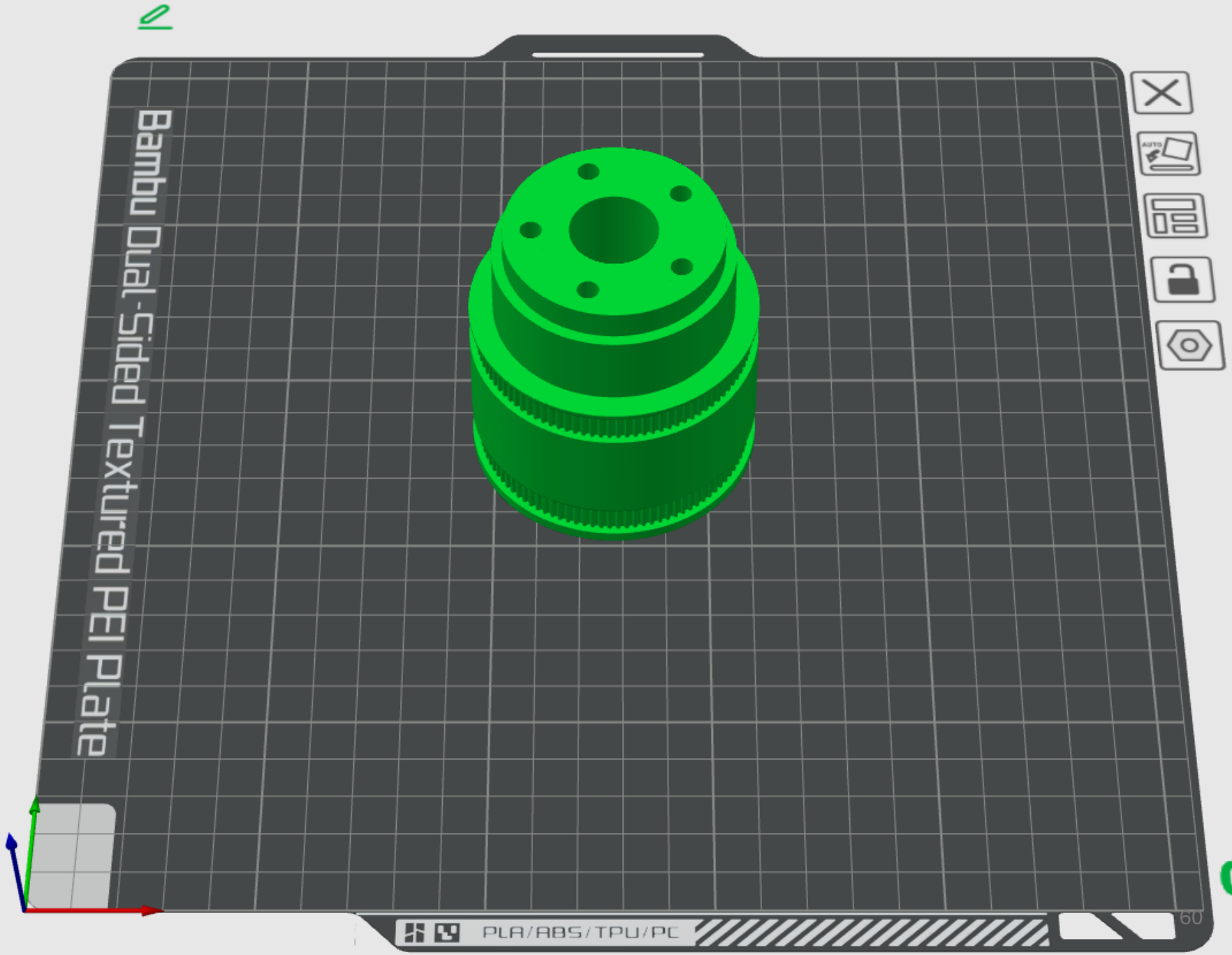


Double gear 100T

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 30%**
- **Wall loops: 5**

Note:
Support is required

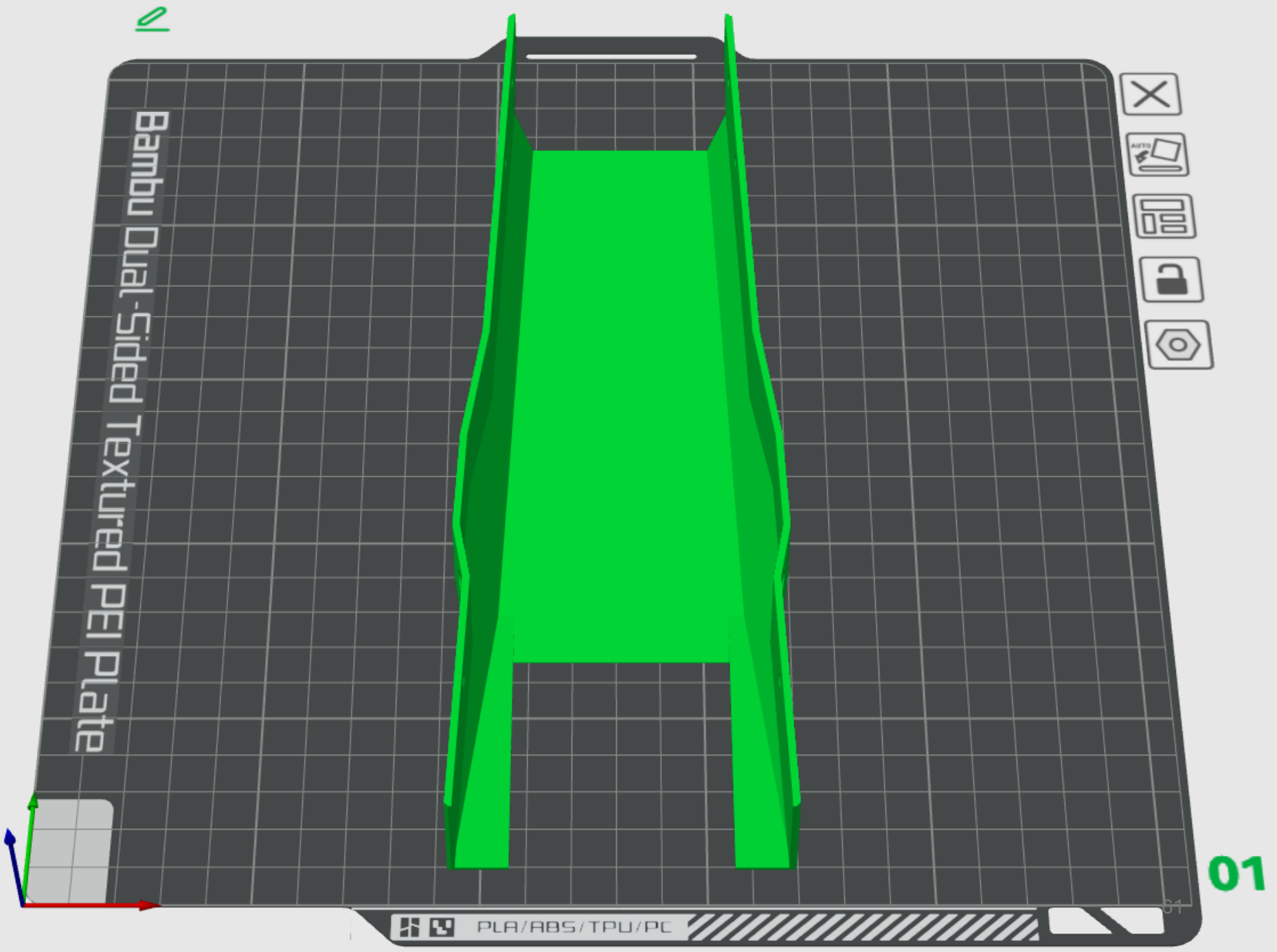


J3 Cover

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

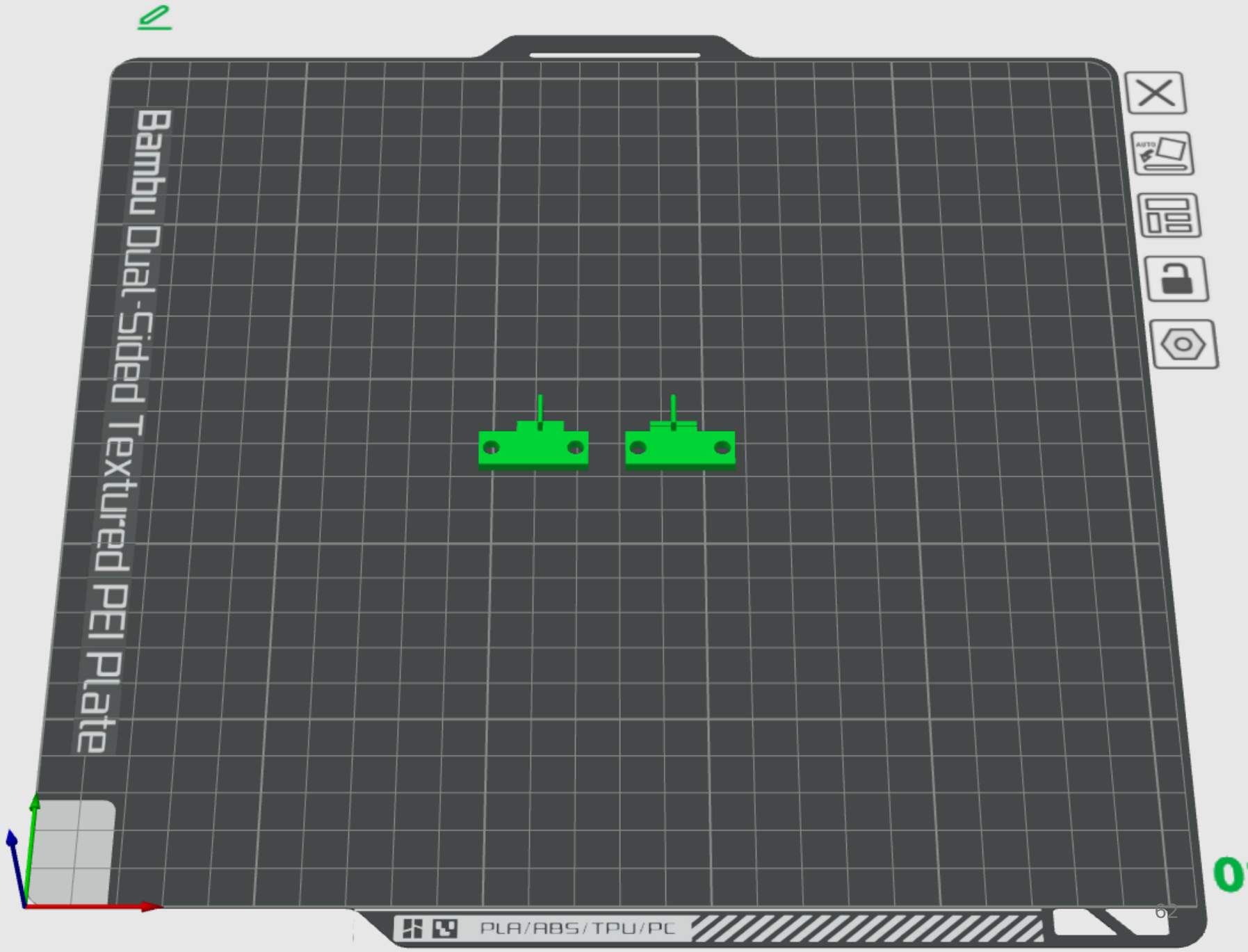
Note:
Support is required



J3 Sensor trigger 1&2

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

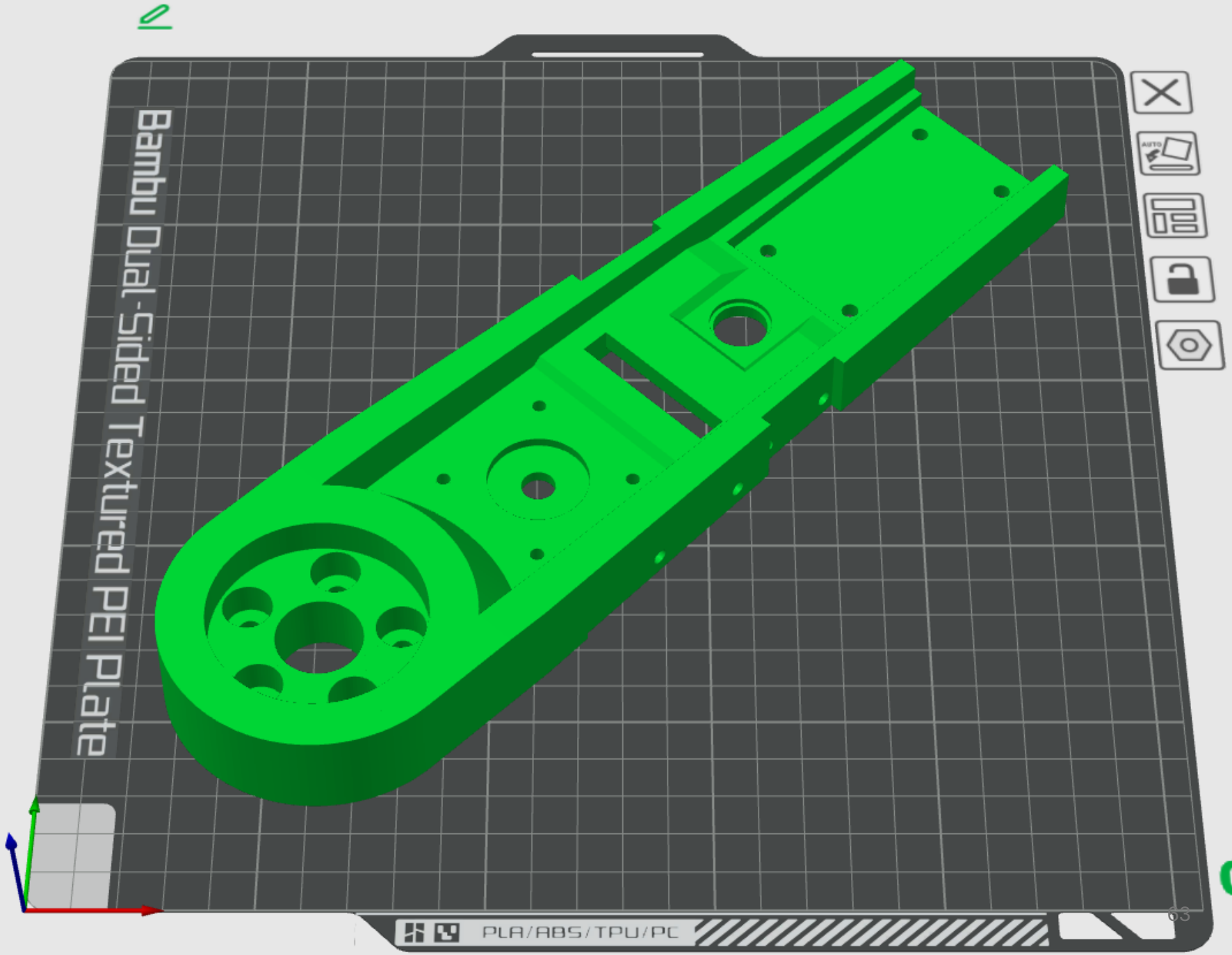


Top link long

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 35%**
- **Wall loops: 5**

Note:
Support is required

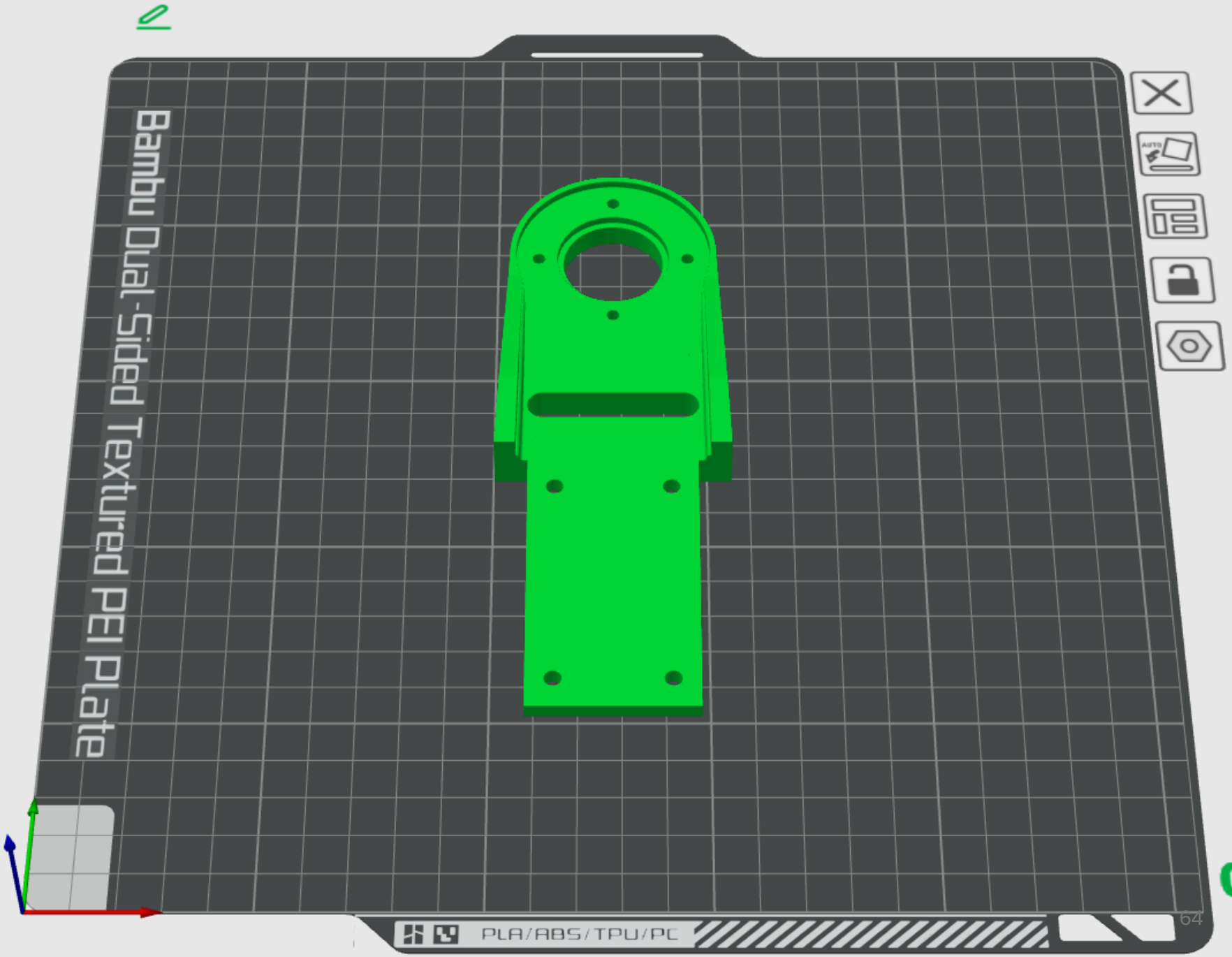


Top link short

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 35%**
- **Wall loops: 5**

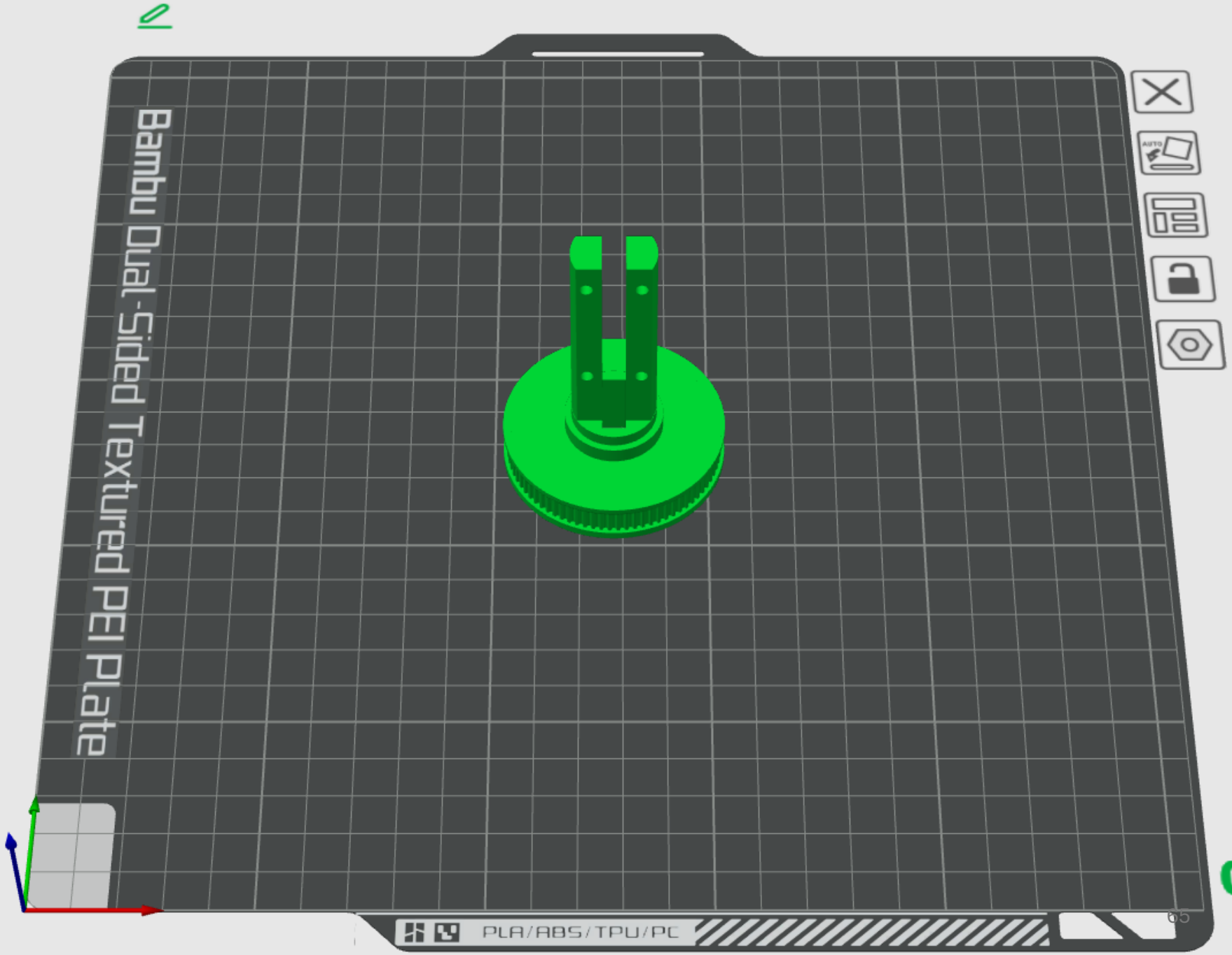
Note:
Support is required



Gear 80T

Print parameters:

- **Material: PETG**
- **Layer height: 0.1mm**
- **Infill: 100%**
- **Wall loops: 4**

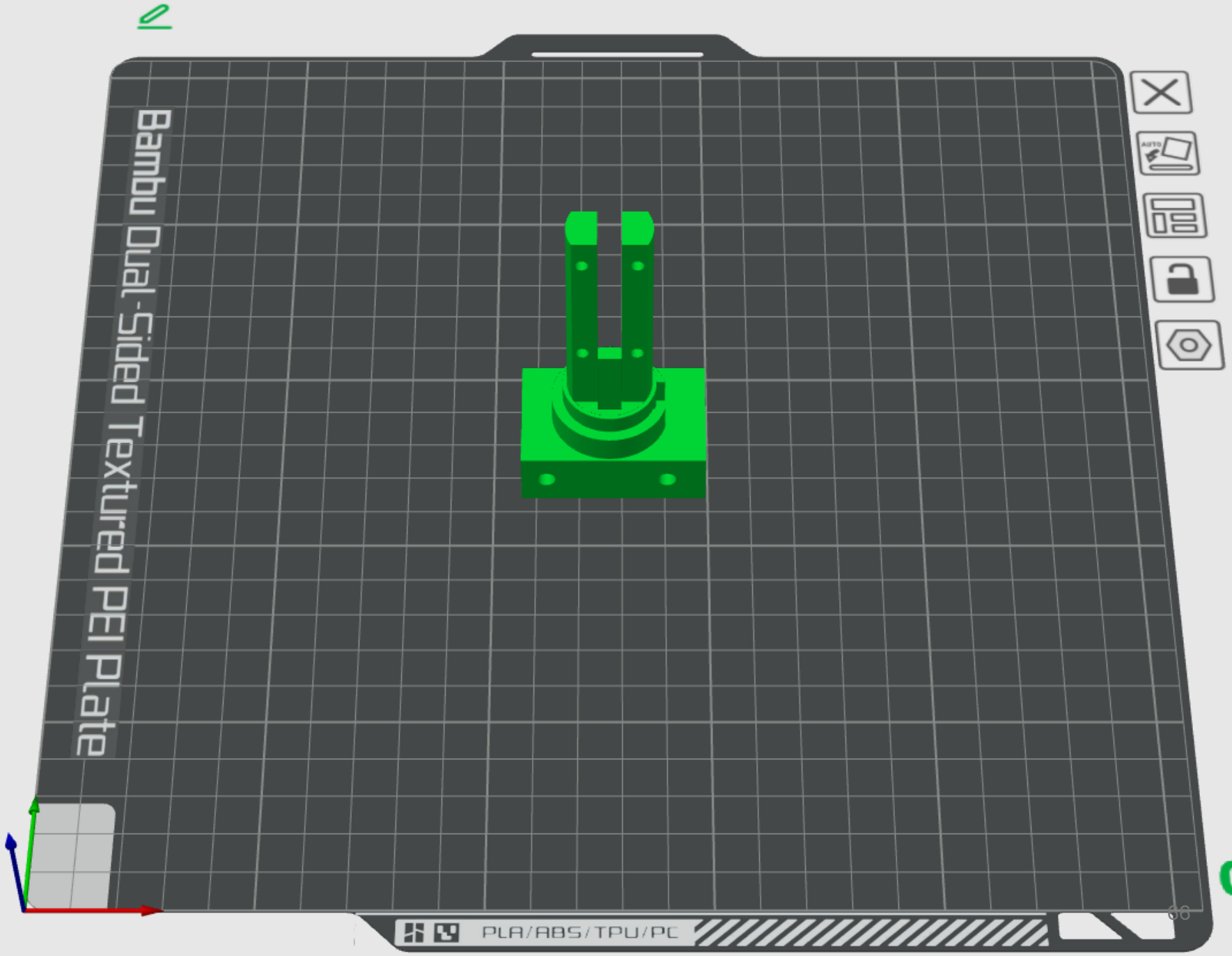


Gripper mount

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 35%**
- **Wall loops: 5**

Note:
Support is required



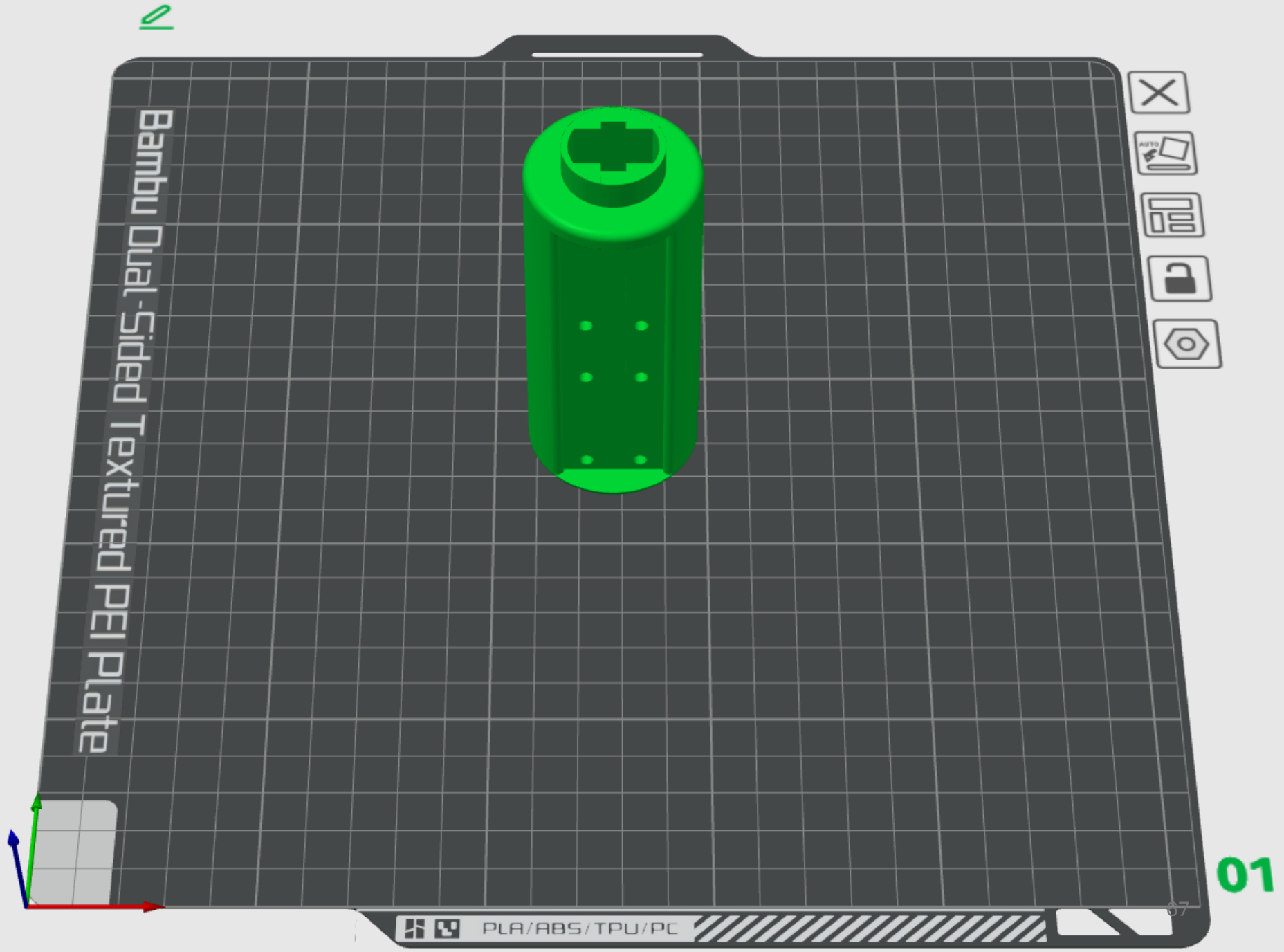
01

J4 Roller

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

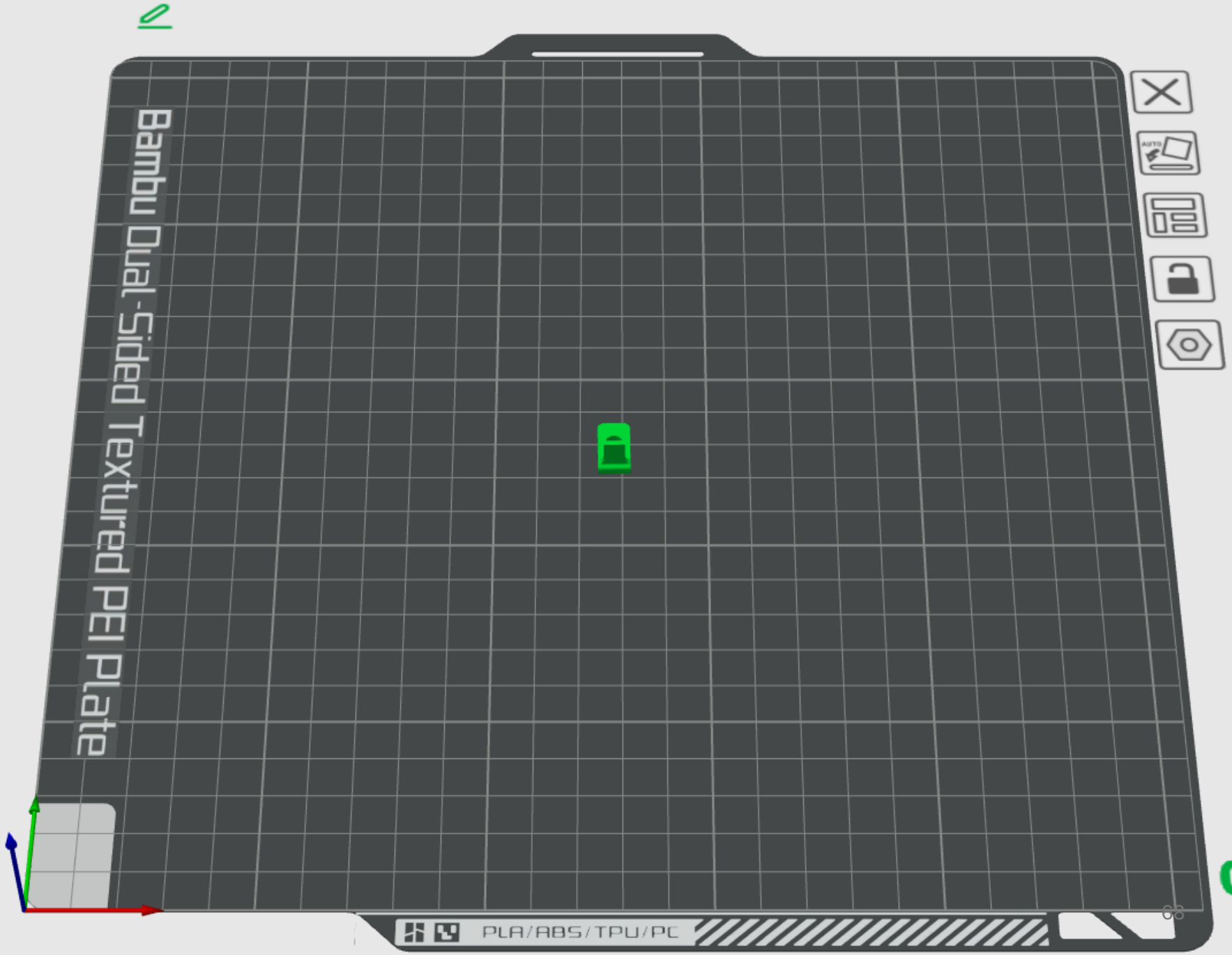
Note:
Support is required



J4 Sensor trigger

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

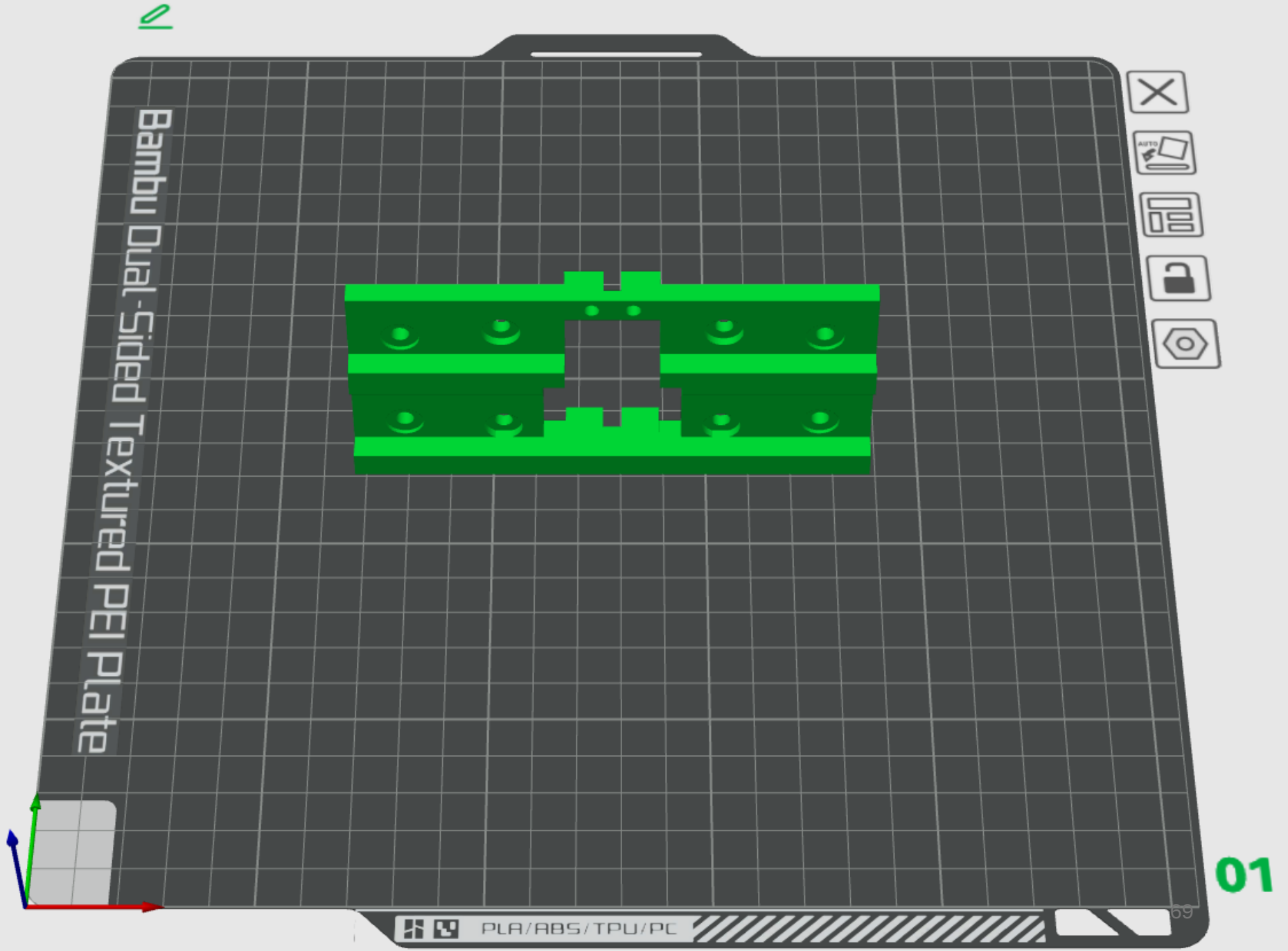


Gripper base

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

Note:
Support is required

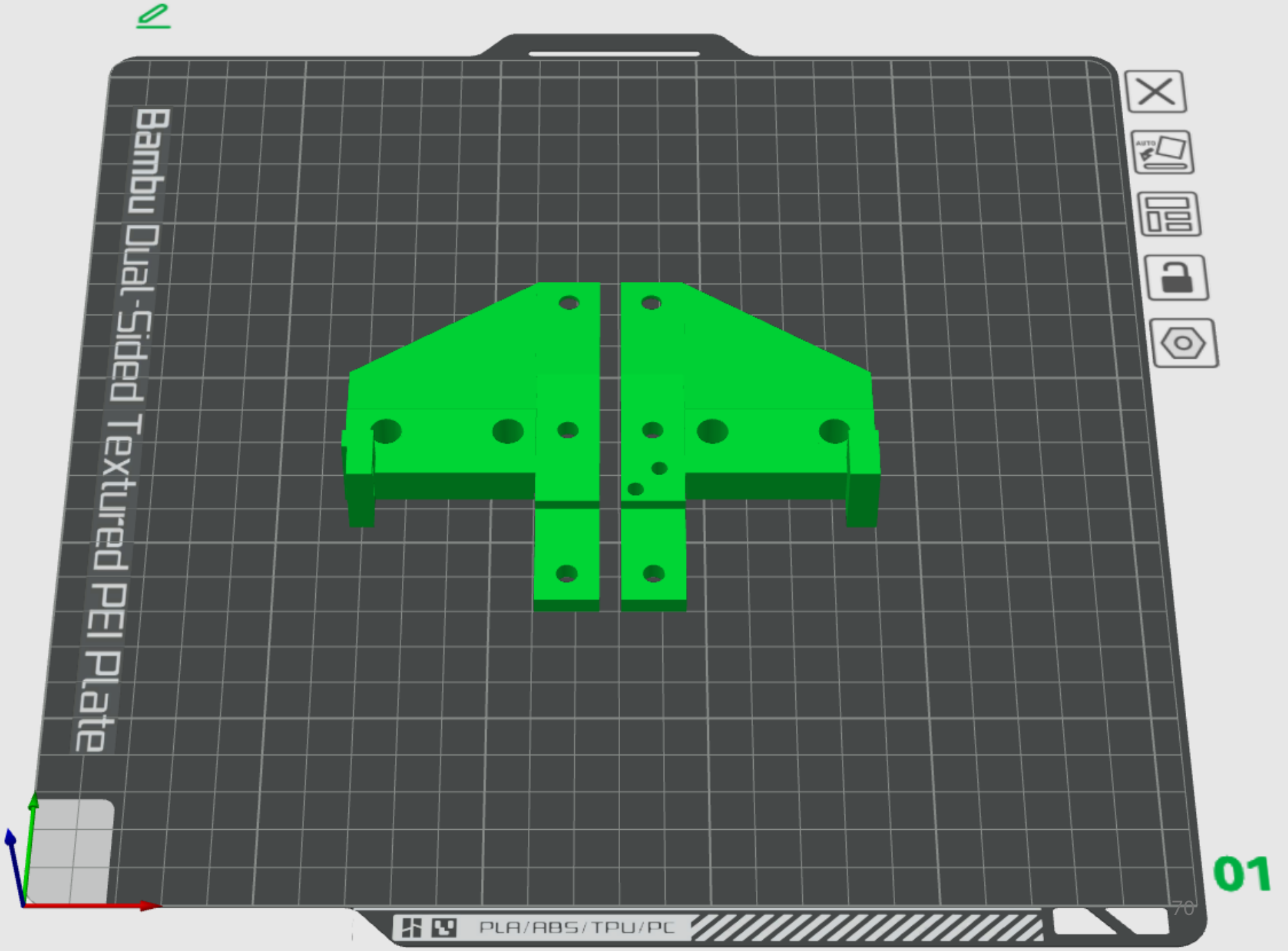


Claw mount right & left

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 20%**
- **Wall loops: 4**

Note:
Support is required

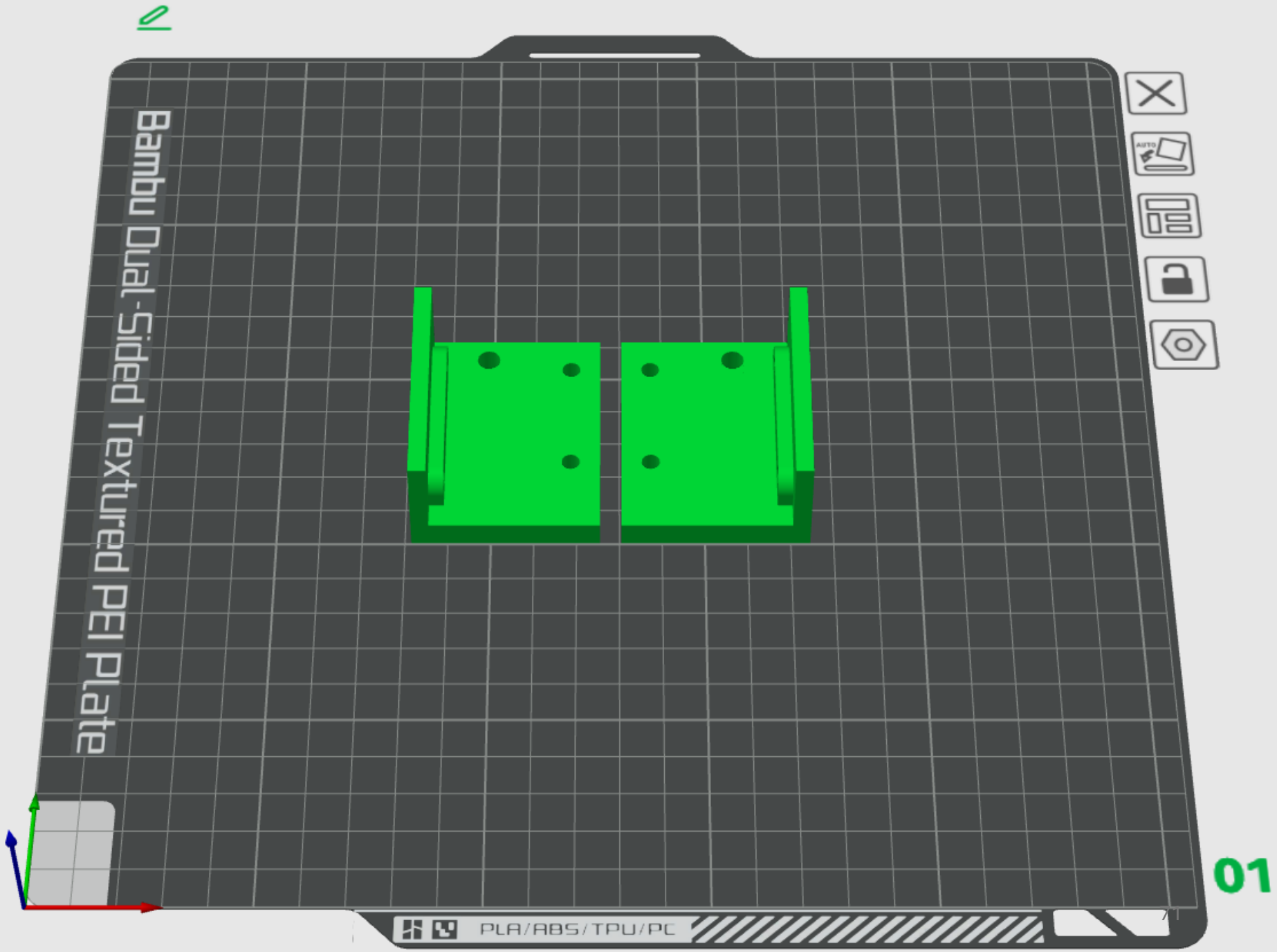


Gripper mount left & right

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 2**

Note:
Support is required

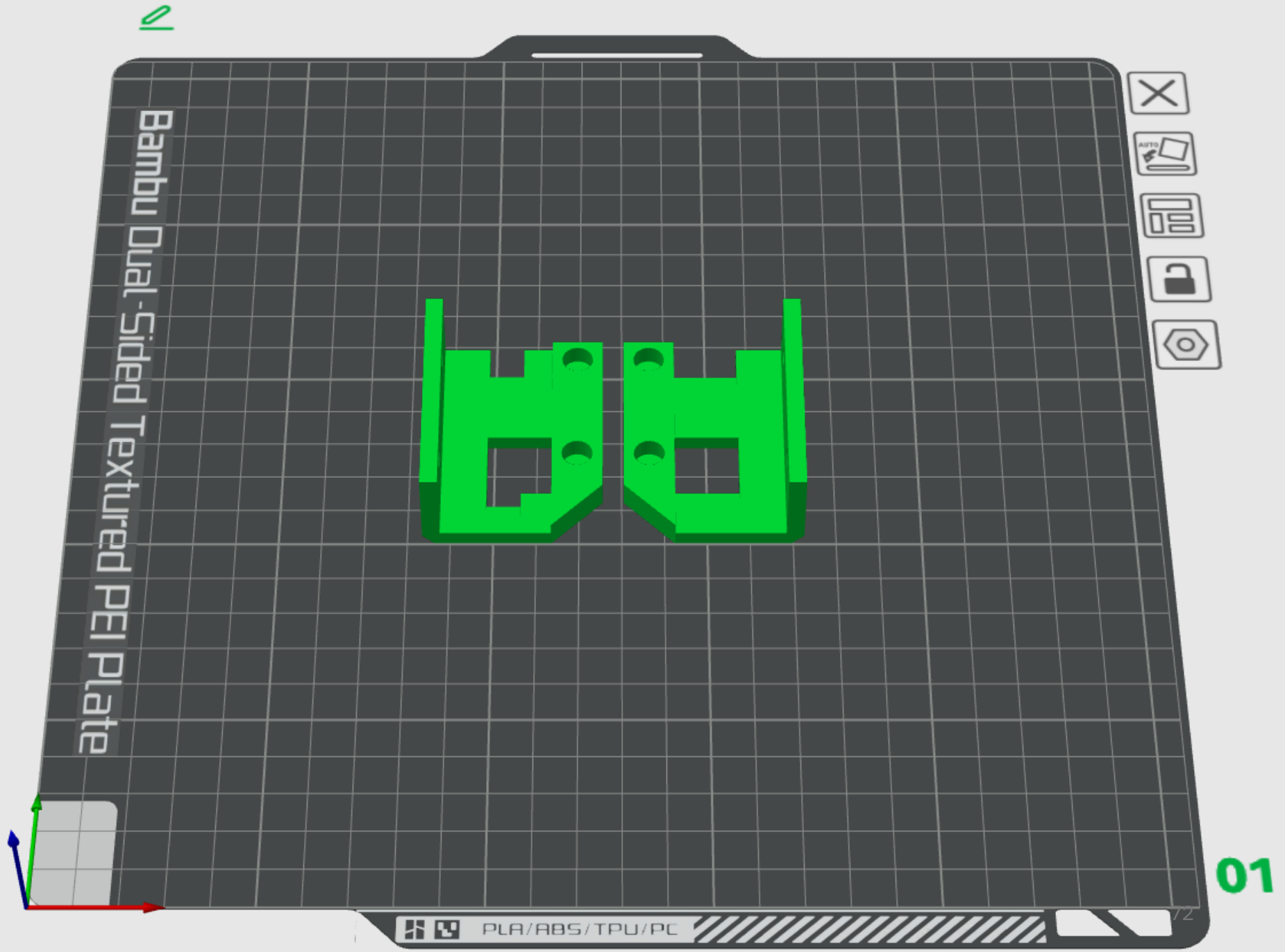


Tube mount left & right

Print parameters:

- **Material: PETG**
- **Layer height: 0.2mm**
- **Infill: 15%**
- **Wall loops: 3**

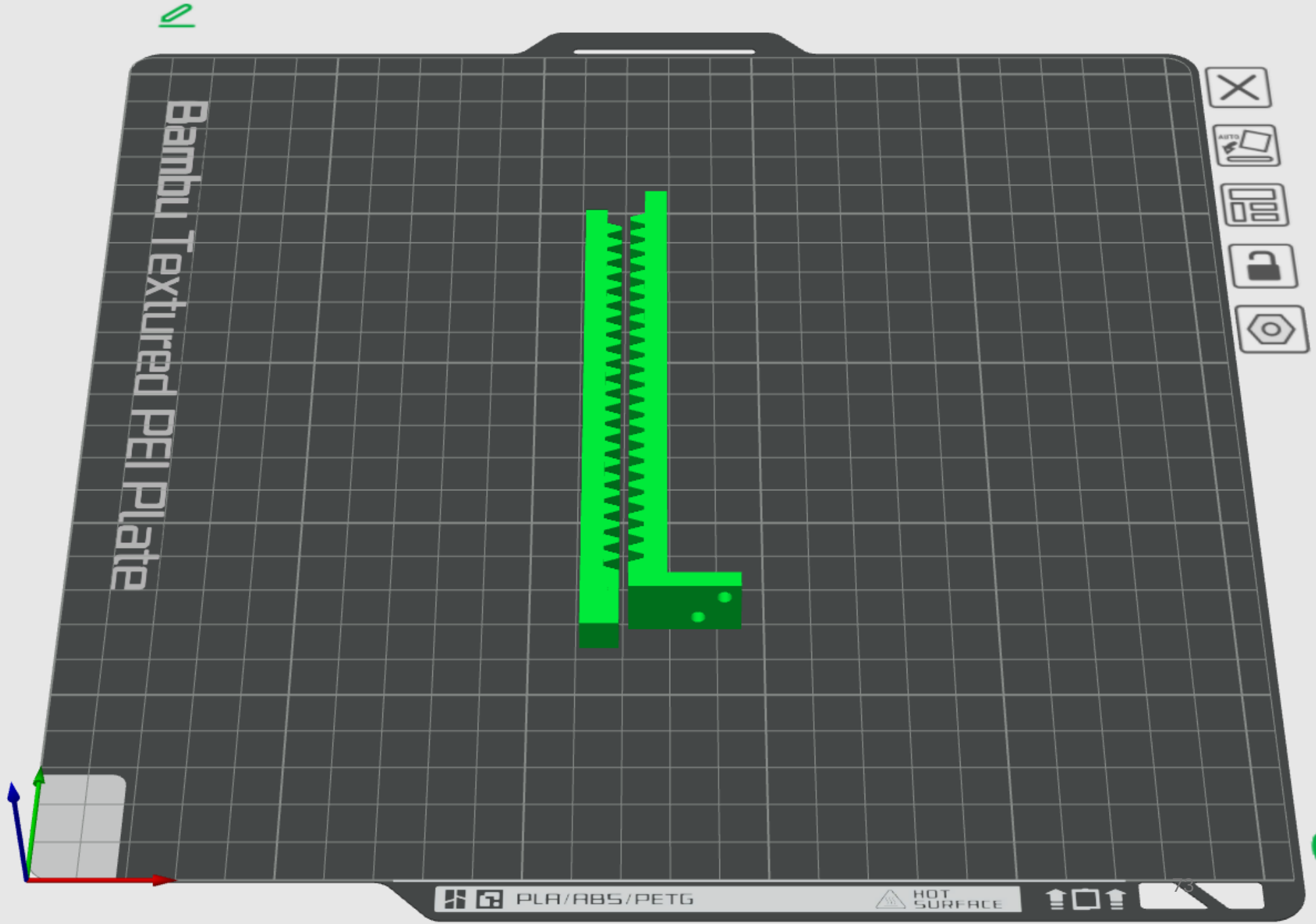
Note:
Support is required



Rack gear left & right

Print parameters:

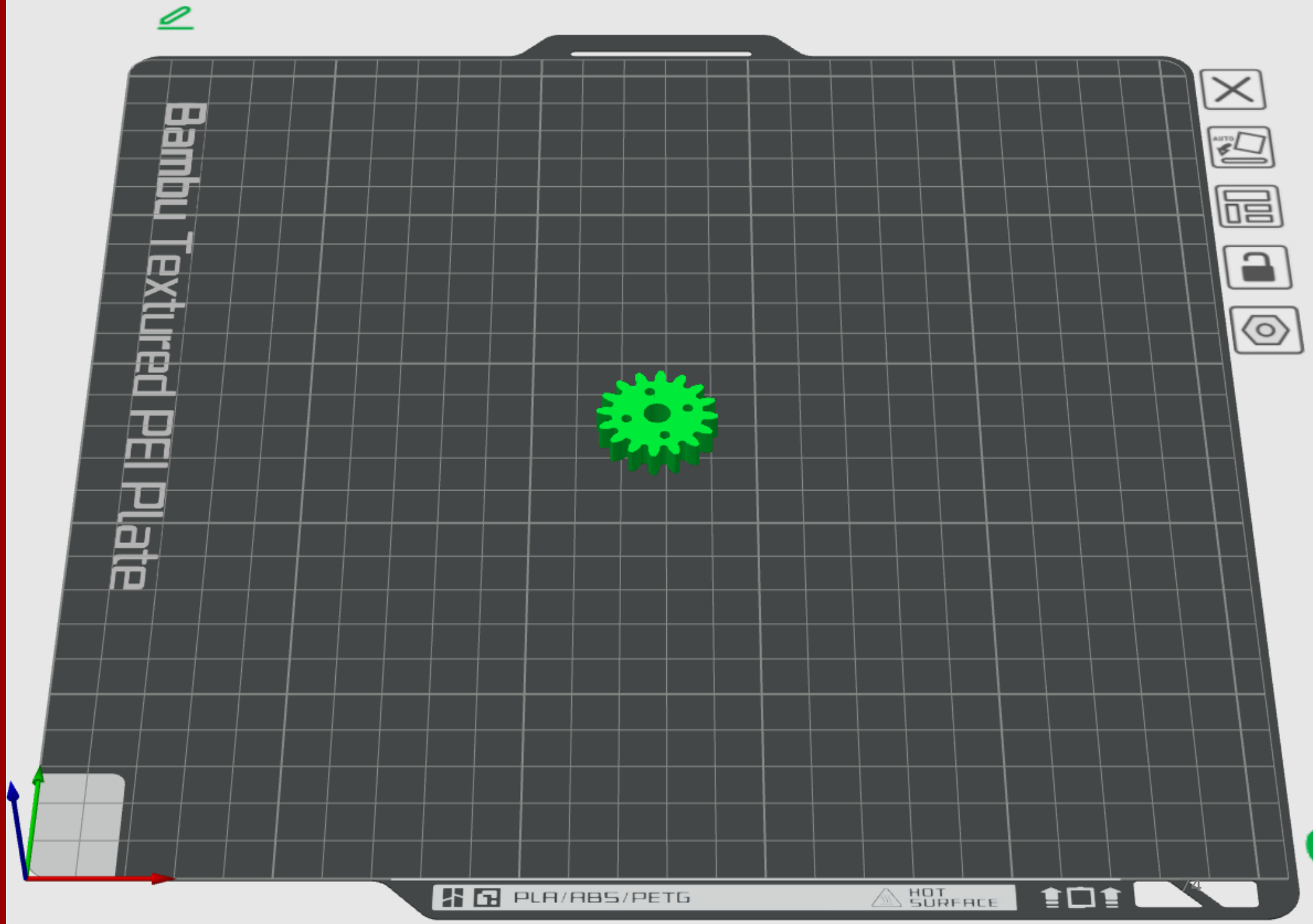
- **Material: PETG**
- **Layer height: 0.1mm**
- **Infill: 100%**
- **Wall loops: 4**



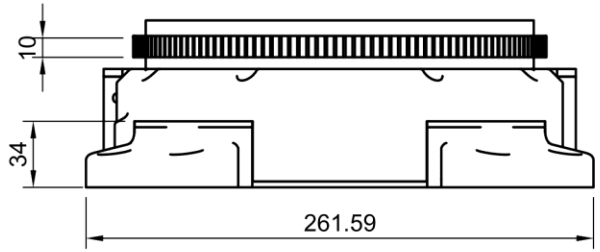
Spur gear 16T

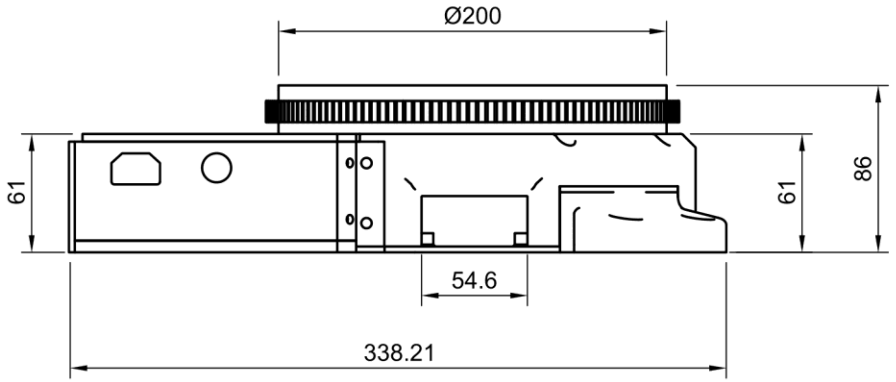
Print parameters:

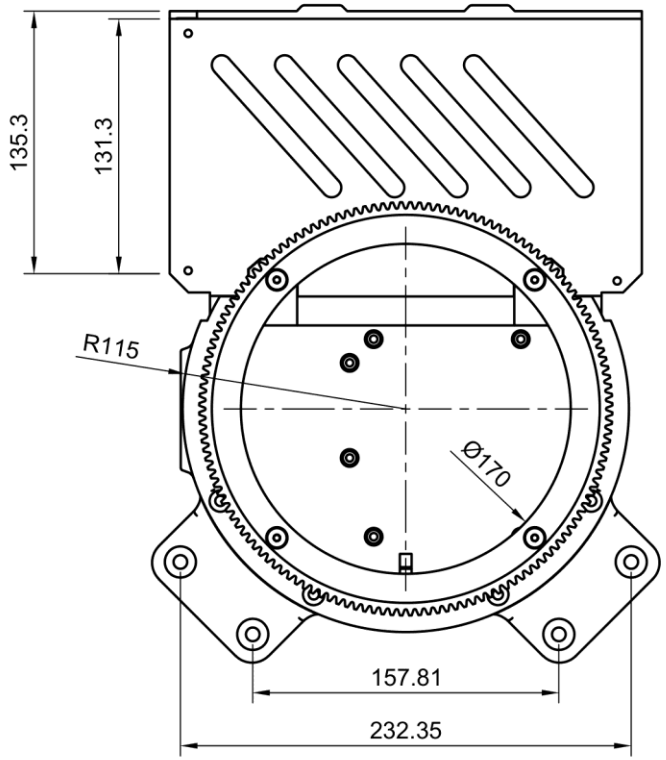
- **Material: PETG**
- **Layer height: 0.1mm**
- **Infill: 100%**
- **Wall loops: 4**

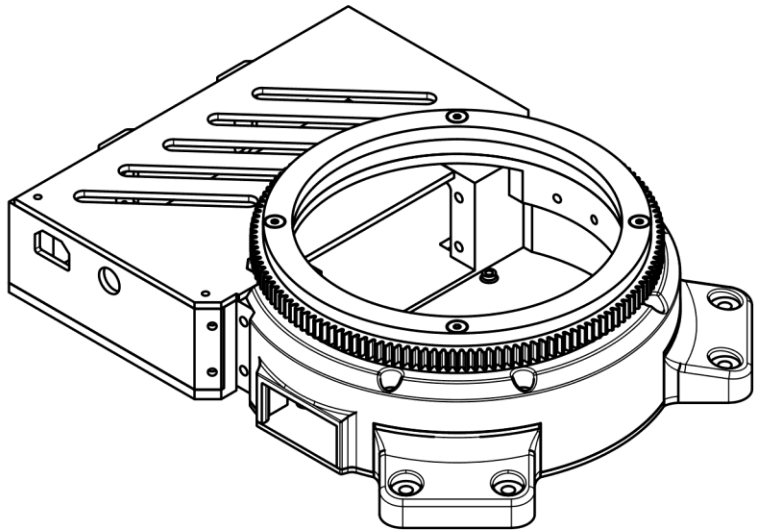


JOINTS

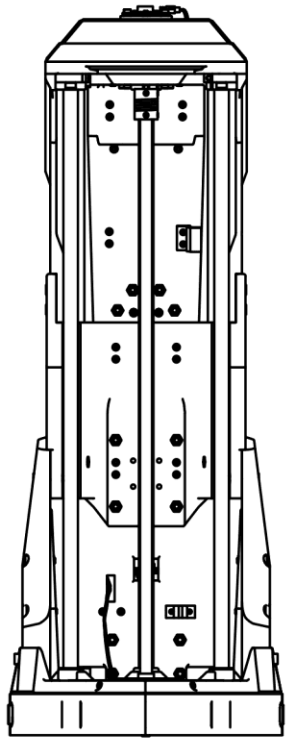


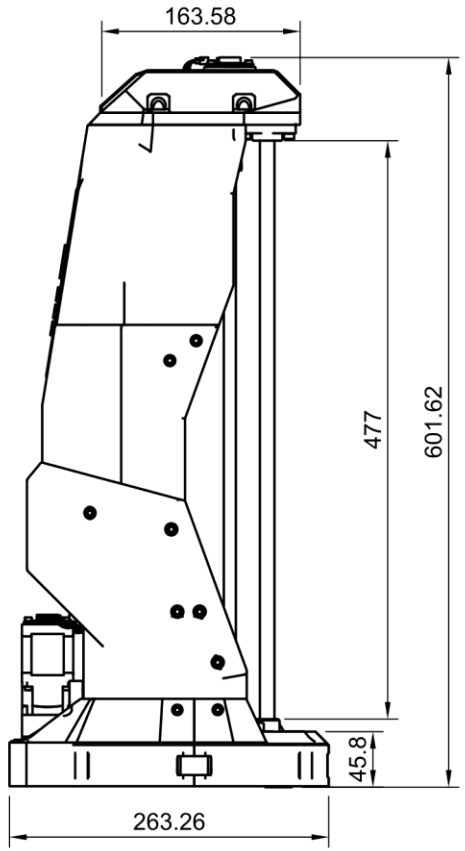


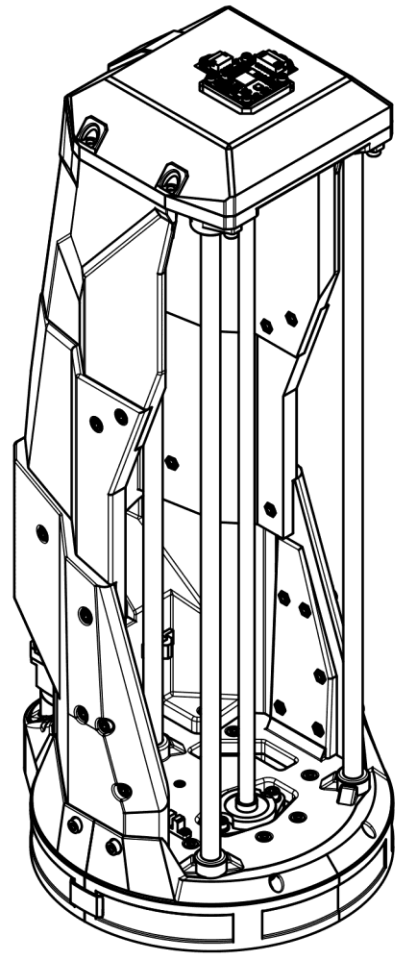


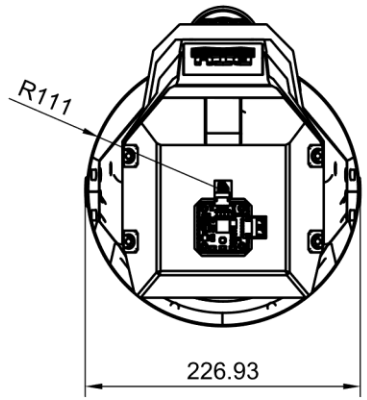


Dept. ELECTRO	Technical reference	Created by Karol Gabor 28.01.2024	Approved by
		Document type TECHNICAL DRAWING	Document status COMPLETED
		Title Robot base DTU SCARA v1	DWG No. 1
			Rev. 1 / Date of issue 28.01.2024 / Sheet 1/1

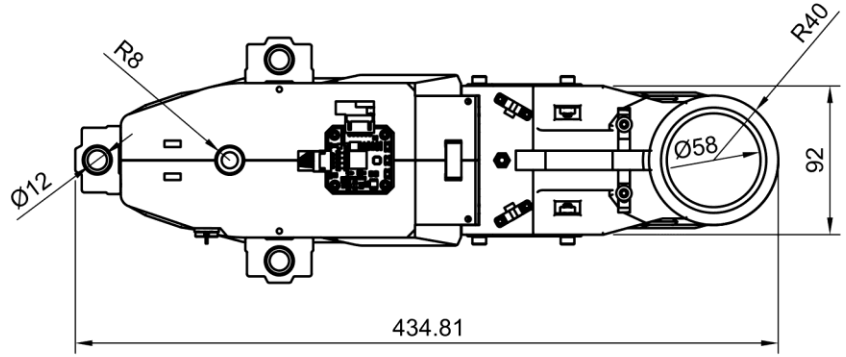


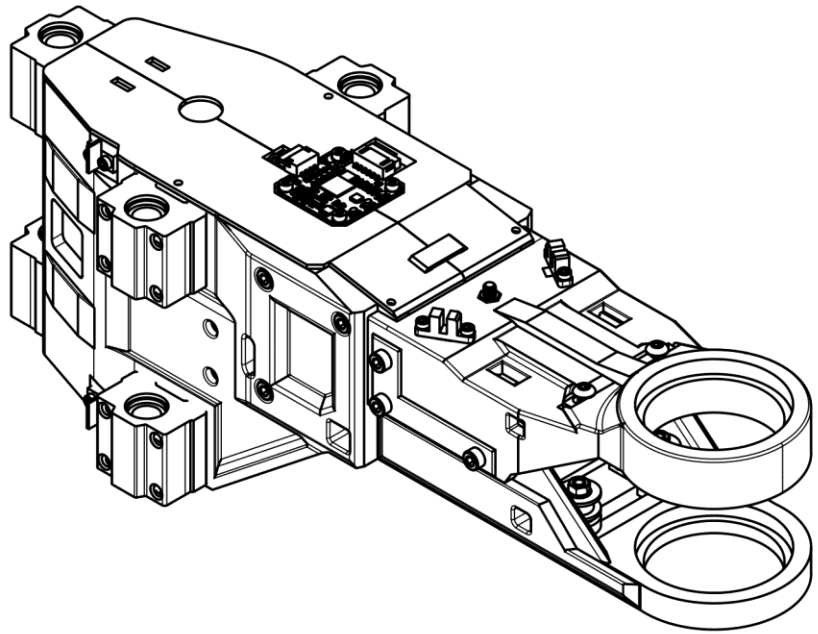


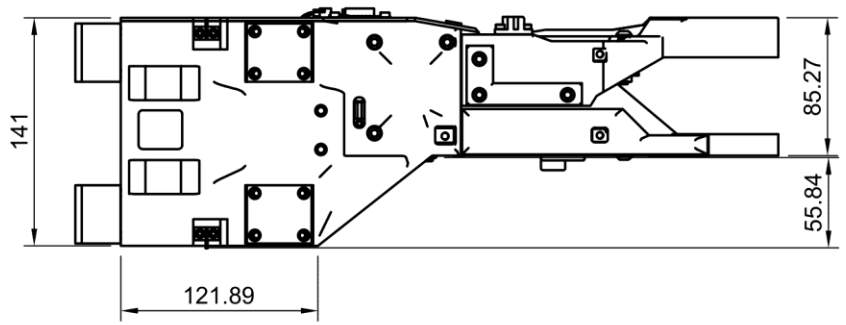


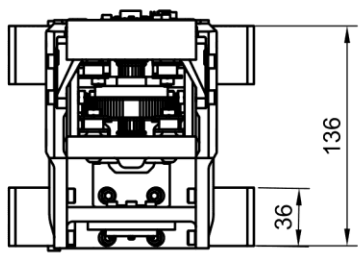


Dept. ELECTRO	Technical reference	Created by Karol Gabor 27.01.2024	Approved by
		Document type TECHNICAL DRAWING	Document status COMPLETED
		Title Joint 1 DTU SCARA v1	DWG No. 1
			Rev. 1 / Date of issue / Sheet 1/1

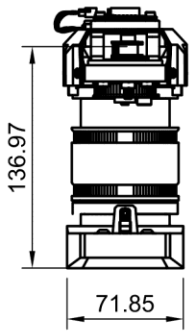


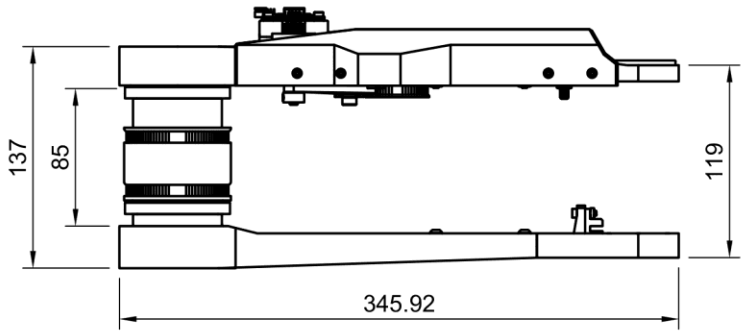


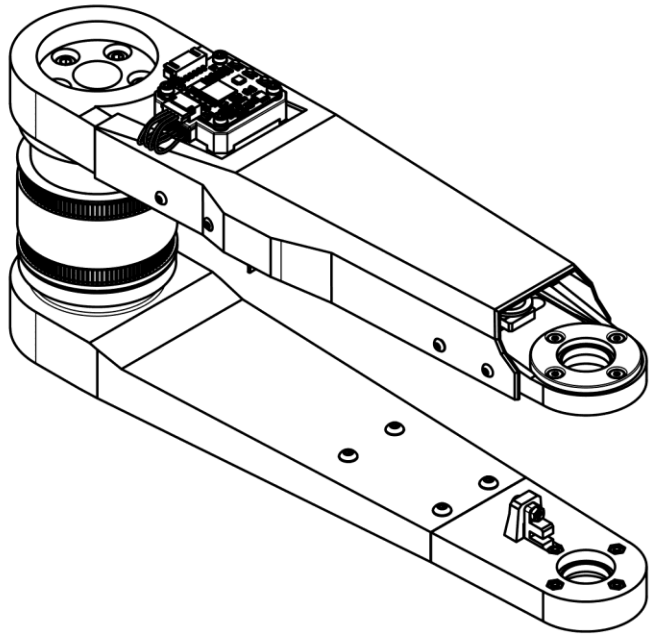


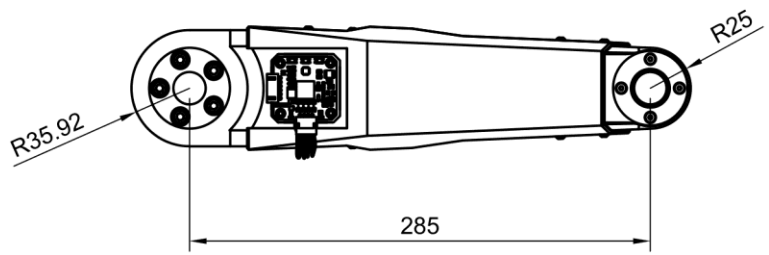


Dept. ELECTRO	Technical reference	Created by Karol Gabor 27.01.2024	Approved by
		Document type	Document status COMPLETED
		Title Joint 2 DTU SCARA v1	DWG No. 1
			Rev. 1 / Date of issue 28.01.2024 / Sheet 1/1

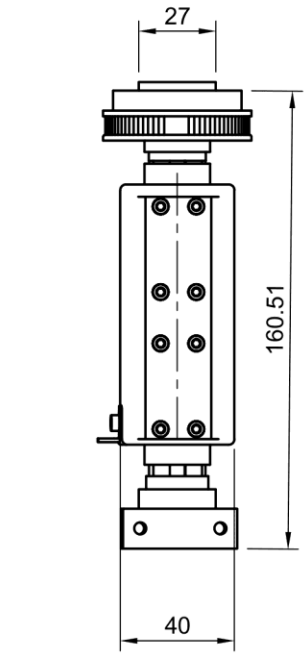


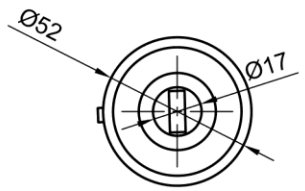


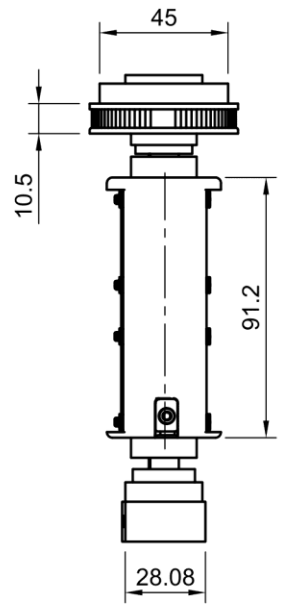


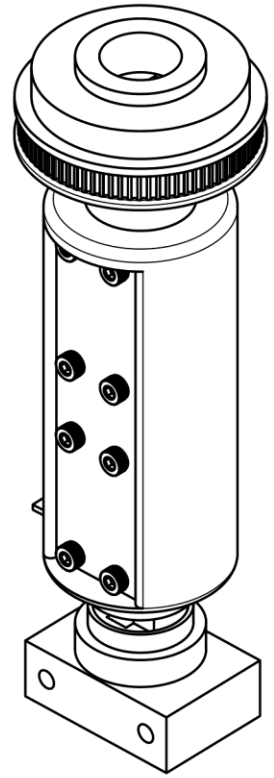


Dept. ELECTRO	Technical reference	Created by Karol Gabor 27.01.2024	Approved by
		Document type TECHNICAL DRAWING	Document status
		Title Joint 3 DTU SCARA v1	DWG No. 1
			Rev. 1 / Date of issue 28.01.2024 / Sheet 1/1

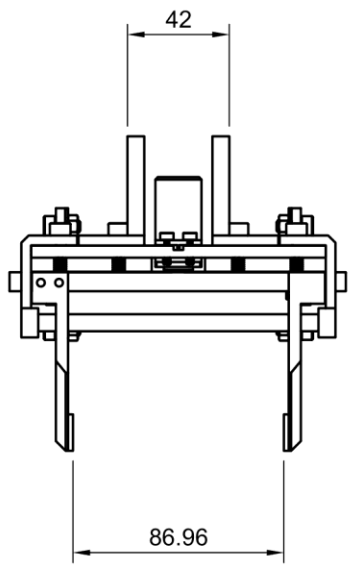


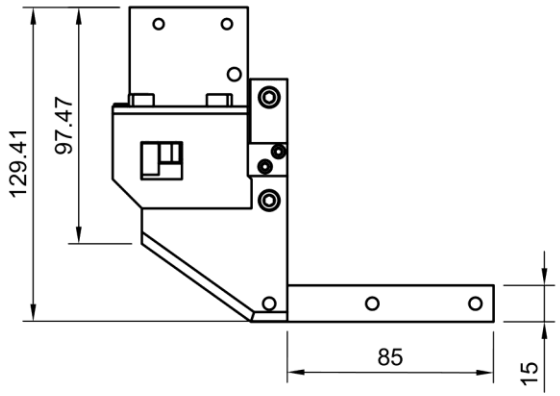


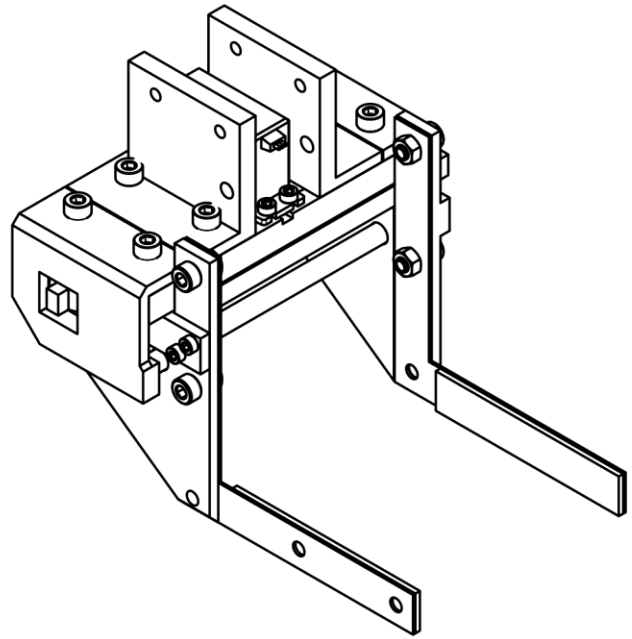


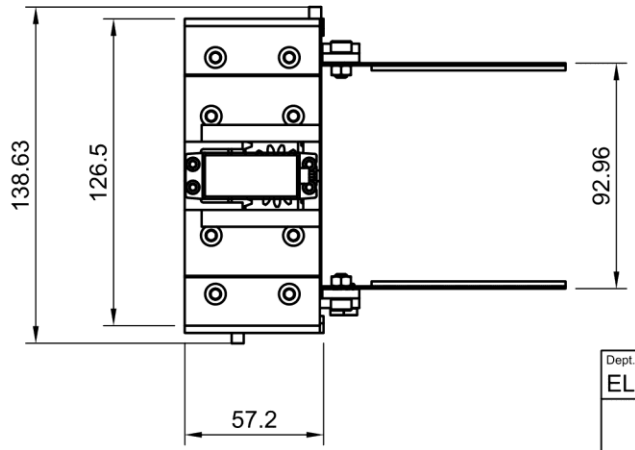


Dept. ELECTRO	Technical reference	Created by Karol Gabor 27.01.2024	Approved by
		Document type TECHNICAL DRAWING	Document status COMPLETED
		Title Joint 4 NEW DTU SCARA v1	DWG No. 1
			Rev. 1 / Date of issue 28.01.2024 / Sheet 1/1









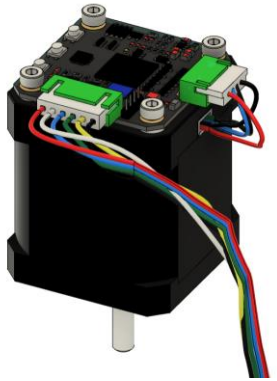
Dept. ELECTRO	Technical reference	Created by Karol Gabor 27.01.2024	Approved by
		Document type TECHNICAL DRAWING	Document status COMPLETED
		Title Gripper DTU SCARA v1	DWG No. 1
			Rev. 1 / Date of issue 28.01.2024 / Sheet 1/1

ACTUATORS

MOTORS

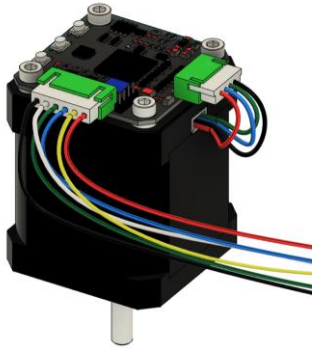
Place the magnet on the motor's shaft rear end (gluing it is good practice to ensure it will stay put). Try to align the magnet on the shaft as precise as possible. Mount the MKS Servo by using M3x10 screws and 3d printed distance washers.

M1 – Joint 1



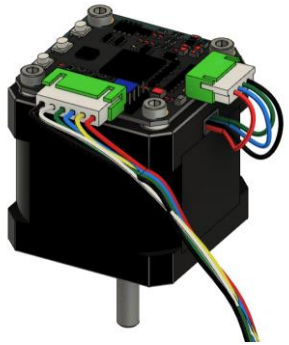
17HS19-2004S1

M2 – Joint 2



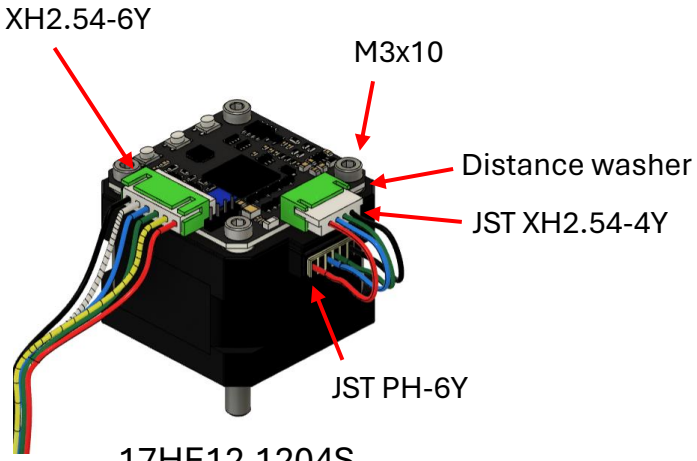
17HS19-2004S1

M3 – Joint 3



17HS15-1504S1

M4 – Joint 4



17HE12-1204S

Phase wiring sequence

A+	A-	B+	B-
Black	Green	Red	Blue

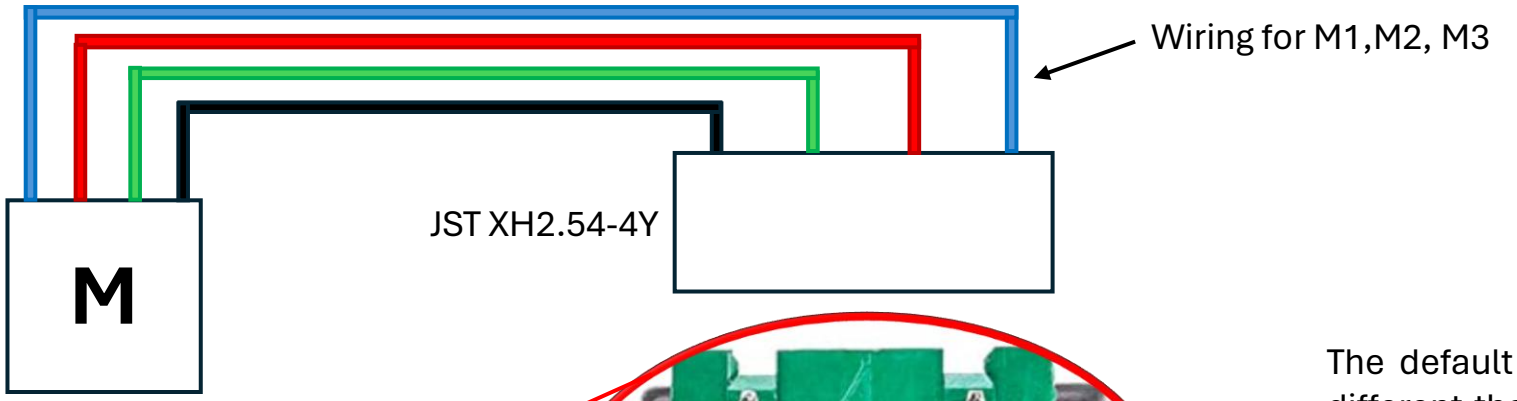
A+	A-	B+	B-
Black	Green	Red	Blue

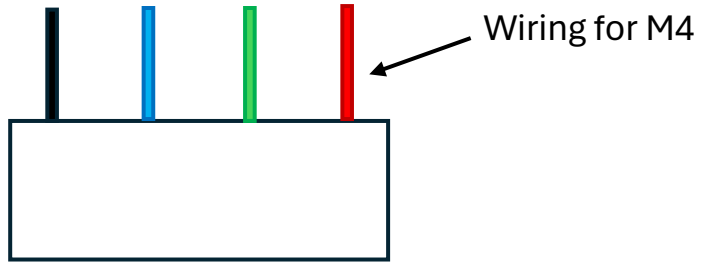
A+	A-	B+	B-
Black	Green	Red	Blue

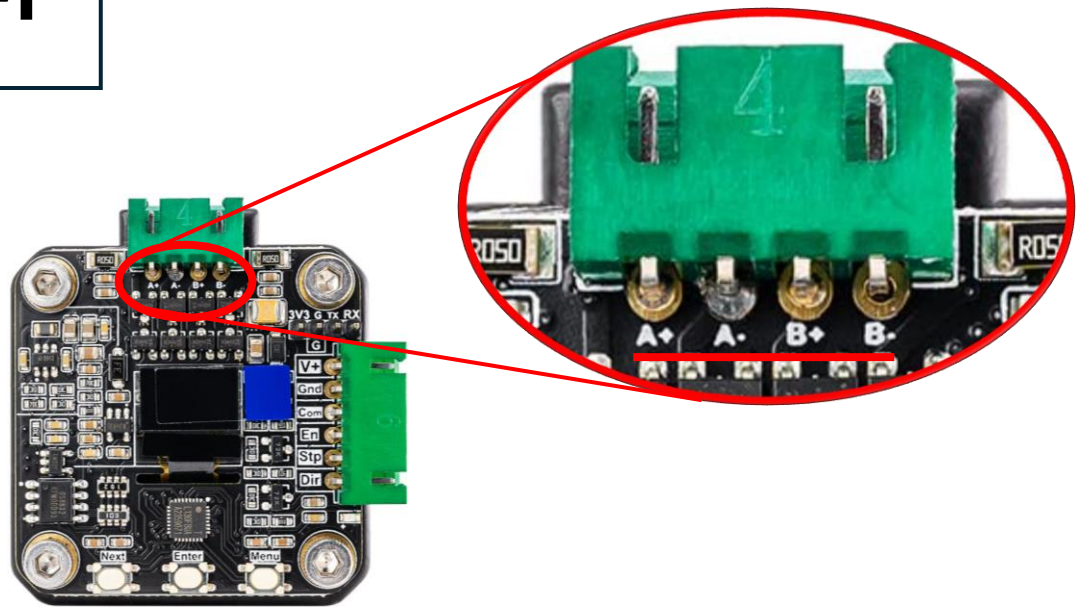
A+	A-	B+	B-
Black	Blue	Green	Red

WIRING SEQUENCE

Connect the motor to the MKS Servo 42C using the JST XH2.54-4Y connector. Refer to the previous page regarding the phase wiring sequence. The motor wiring is an important step that has to be checked before assembling the whole robot.







MKS Servo42C

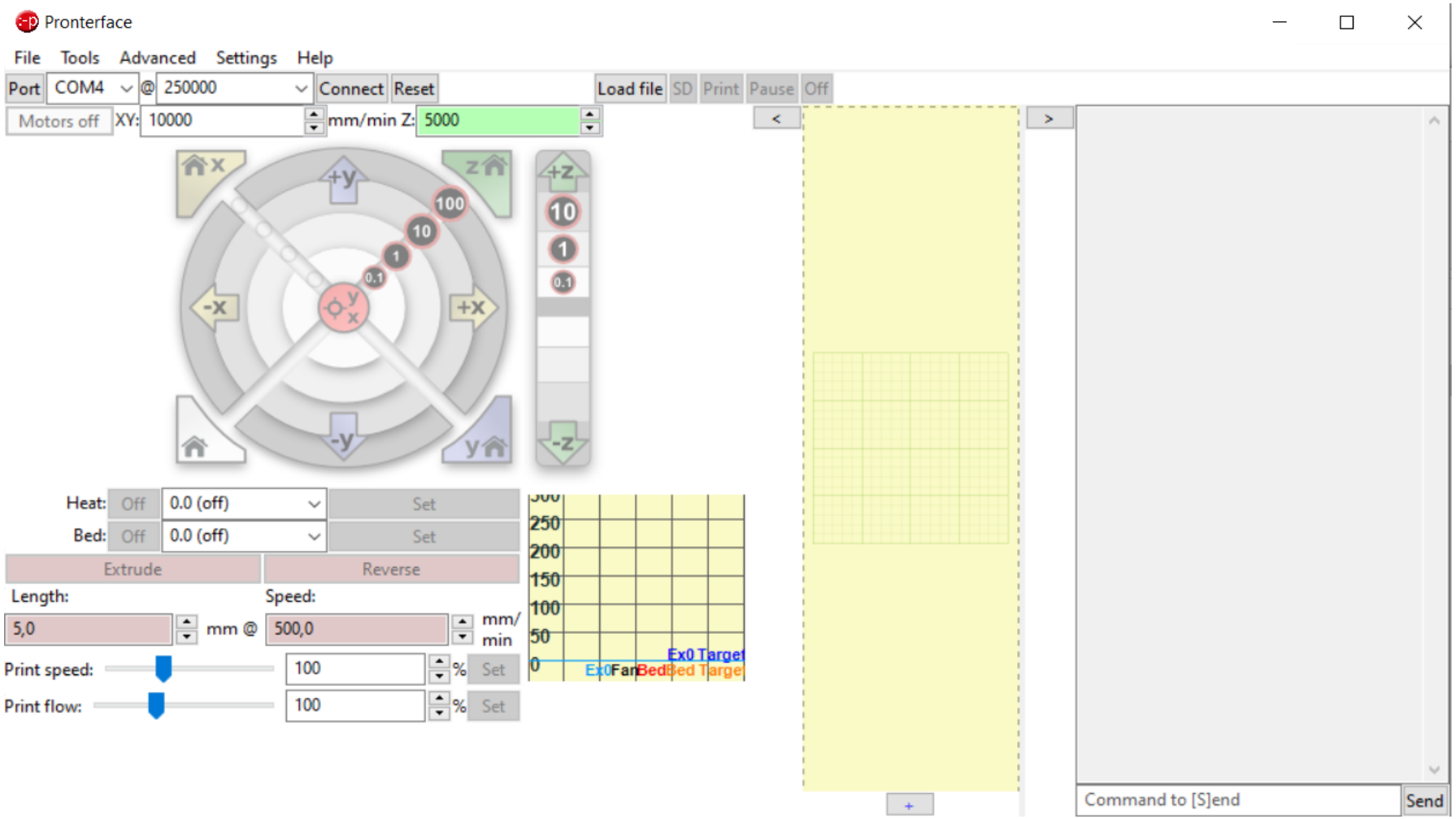
The default wiring sequence for the motor used in joint 4 is different than in the other motors. Use the cable provided for this motor.

To make sure the wiring is correct, you can measure the resistance between the wires after it has been connected to the motor. In a bipolar stepper motor, the ends of the coil are represented as two pairs – A+ and A- as well as B+ and B-. The measured resistance between the two ends of each pair should be 0 Ω.

An incorrectly wired motor will result in a short when powered. This might be indicated by the motor not being able to move, very high current draw and a buzzing sound.

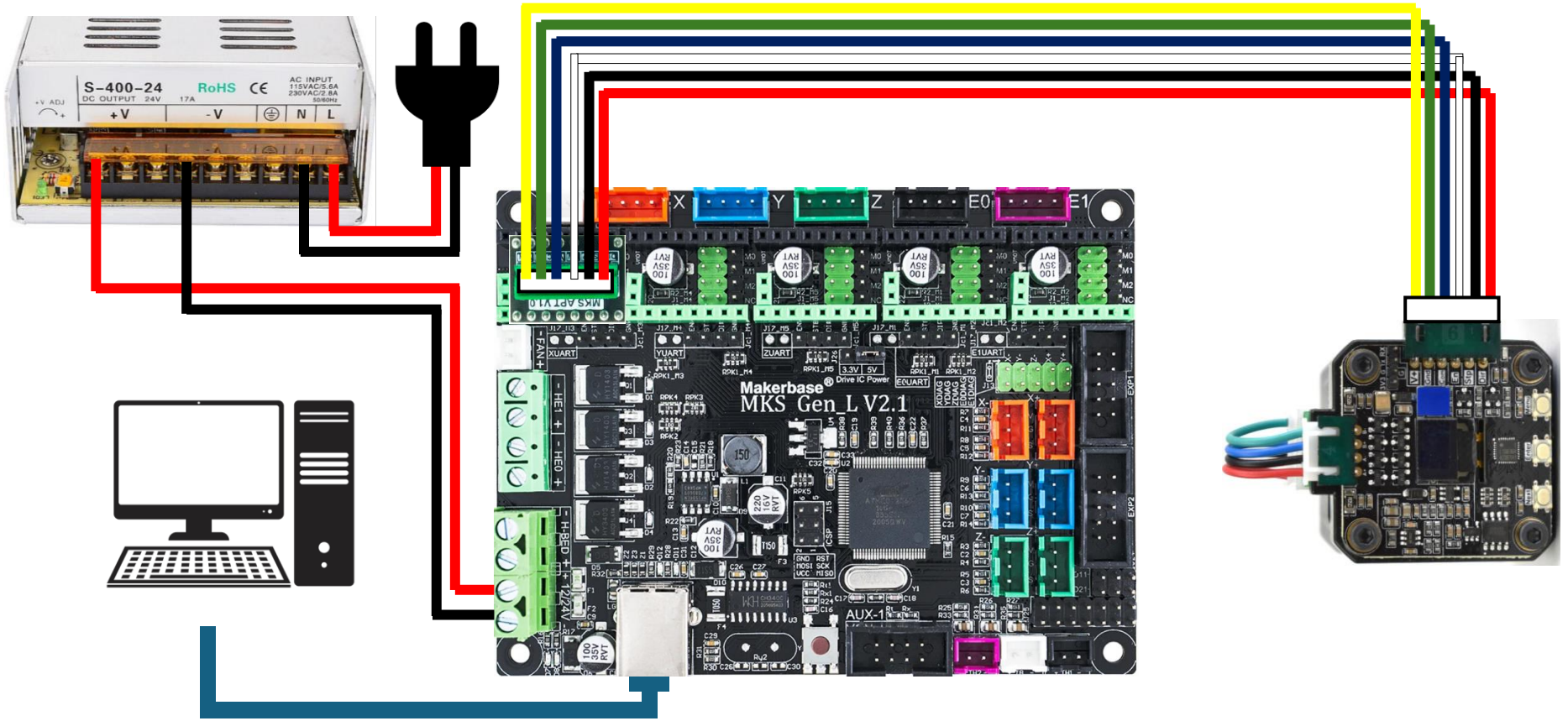
PRONTERFACE

To verify if the motors have been wired up correctly, the Pronterface software can be used. It is a simple software that allows for basic motor controls. Using this UI, it is possible to check if the motors are rotating both directions.



WIRING FOR MOTOR TEST

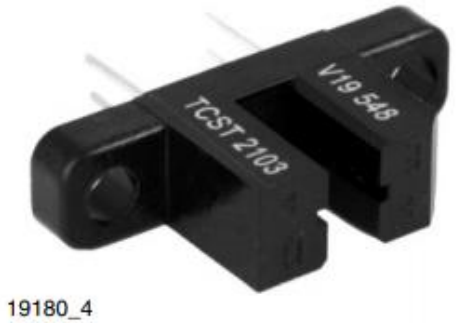
Attach the MKS Servo42C to the motor and then connect it to the MKS Gen L2.1 board. Next, connect the MKS board with a PC using a USB cable (USB B port on the MKS board). Set the baud rate to 250000 and specify the port. Then, try to move the motor using the UI. Use the following wiring method. You can use the robot's power supply (also requires a power cord with bare wires) however any 12V or 24V power supply also works (should be able to provide at least 1A of current).



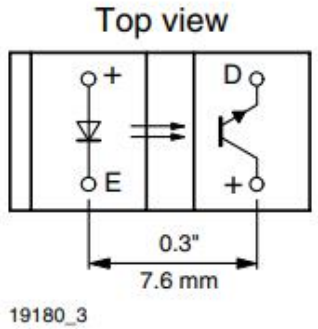
ENDSTOPS

ENDSTOPS

Optical endstops TCST2103 have been used for all joints. Joint 2 and joint 3 have two endstops each. They are placed on the edges of the motion range. Joint 1 and joint 4 have only one endstop each. That is because the movement of these joints is not limited by design, allowing for a full 360° rotation. ***NOTE*: EVEN THOUGH THESE JOINTS CAN ROTATE 360° THEY ARE STILL LIMITED BY THE CABLES CONNECTED TO A FIXED PLACE – THE MOTHERBOARD. ROTATING THESE JOINTS CONTINUOUSLY FOR MORE THAN ONE ROTATION CAN LEAD TO THE CONNECTORS BEING PULLED OUT OF THE MOTHERBOARD OR PERMANENT DAMAGE TO THE WIRES**



19180_4

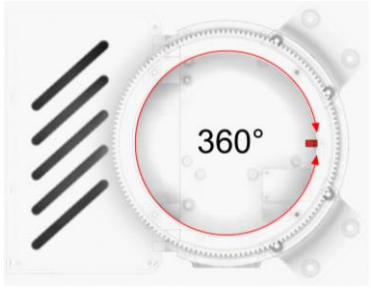


19180_3

DESCRIPTION

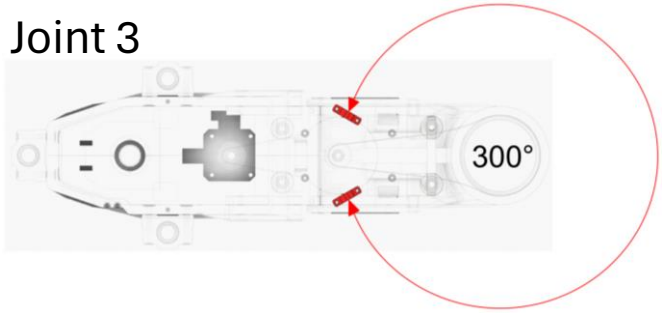
The TCST2103, TCST2202, and TCST2300 are transmissive sensors that include an infrared emitter and phototransistor, located face-to-face on the optical axes in a leaded package which blocks visible light. These part numbers include options for aperture width.

Joint 1

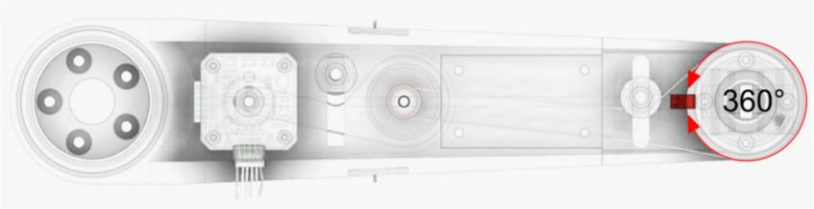


Joint 2

Joint 3

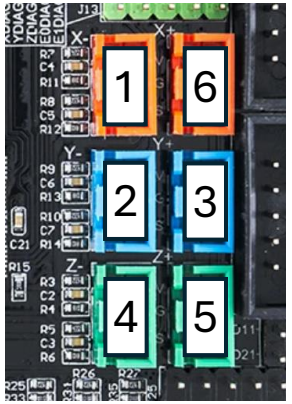


Joint 4

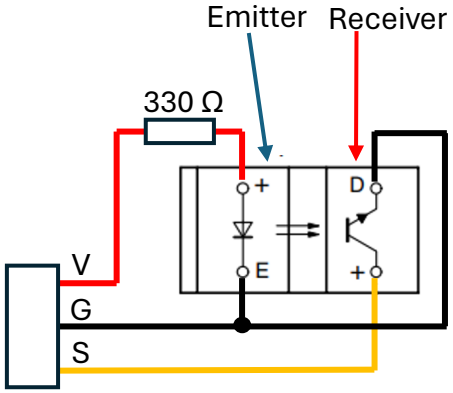


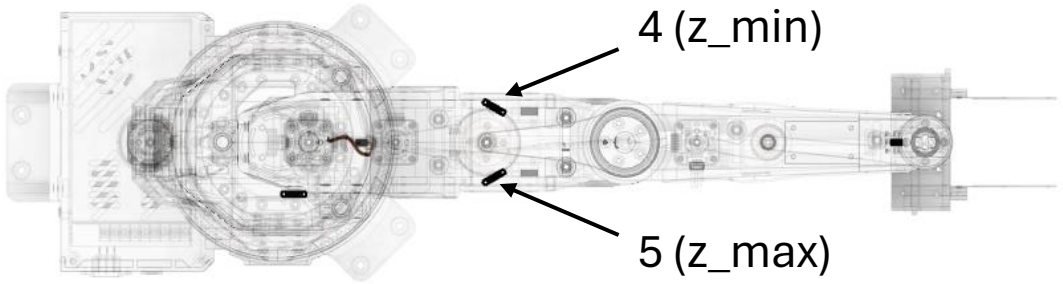
ENDSTOPS

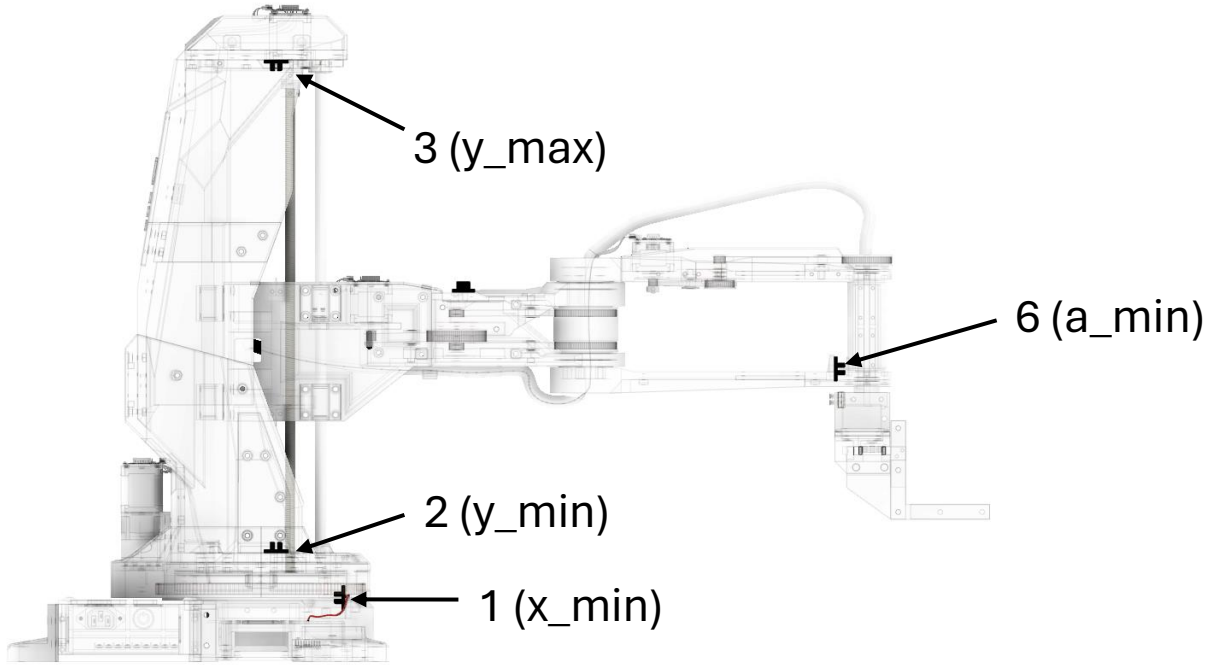
MKS Gen L2.1 allows for up to 6 endstop connections (refer to the motherboard description and pinout schematic). The default state for these sensors is “close” until something blocks the emitted infrared light. This means that the “open” state corresponds to a triggered endstop. When the state is “closed”, the current flows between S and G and the board recognizes the state. Consequently, an “open” state means that there is not current drawn from the S pin (there is no direct short between 5V and GND due to the built-in resistors, you can read more about it by checking the MKS Gen L2.1 board schematics).



JST 2.54 XH-3Y connectors





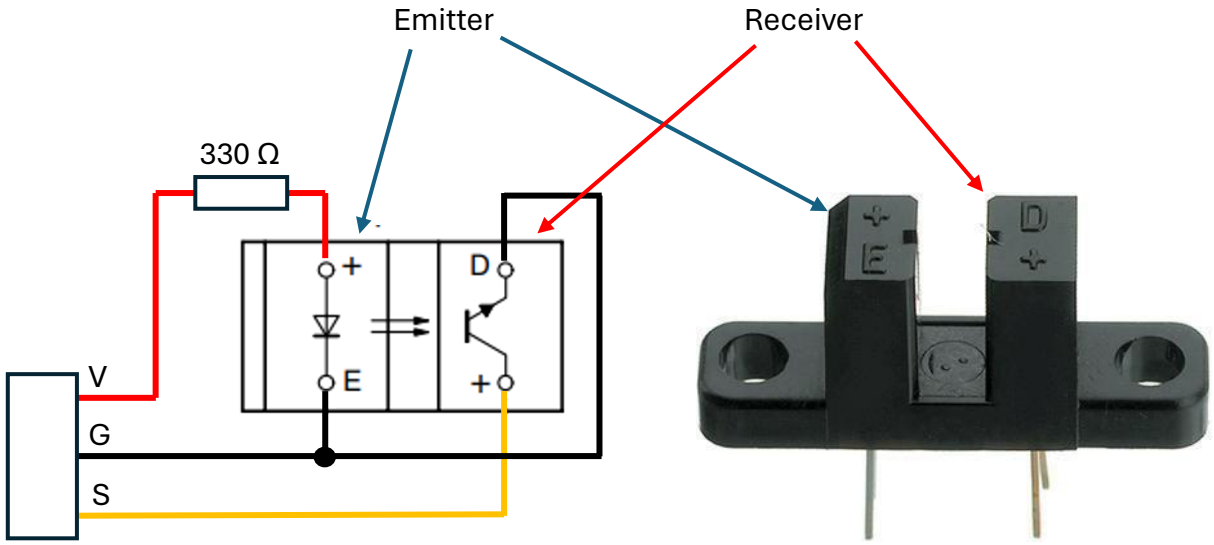


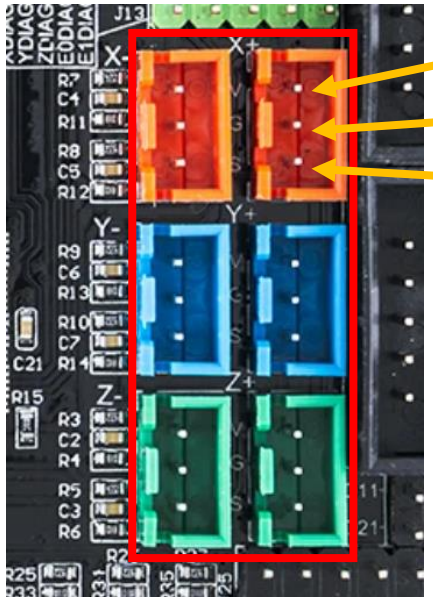
ENDSTOP WIRING

The emitter requires a 330 Ω resistor on the V input. Additionally, to reduce the amount of wires running through the joints, the two GND sides of the emitter and receiver can be connected, allowing for 1 wire instead of 2. All 6 endstops must be connected to the board.

The MKS board requires 2 connections (for a simple switch endstop) or 3 connections (for sensors like the TCST2103 photo interrupter).

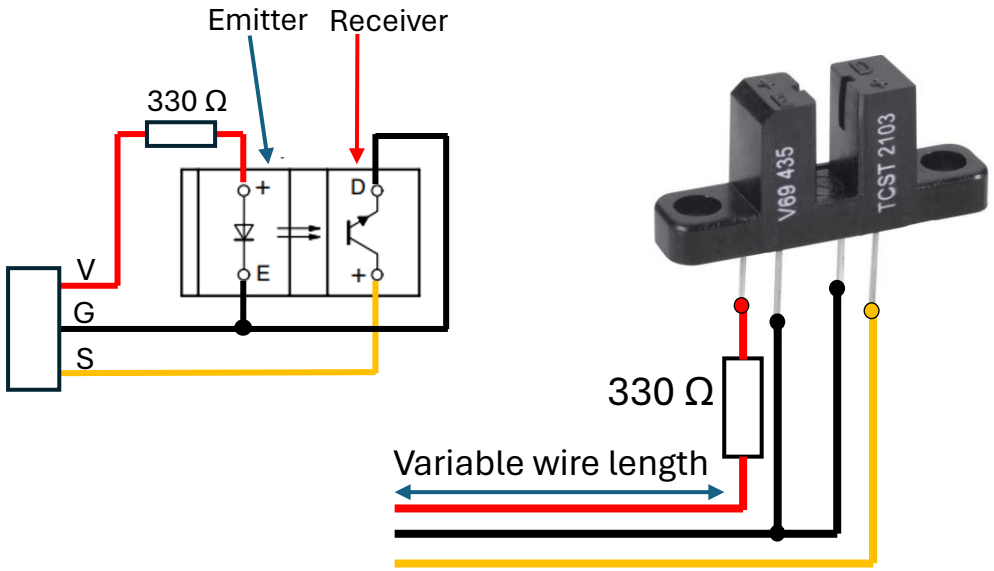
The 3 connections are: V, G, S.





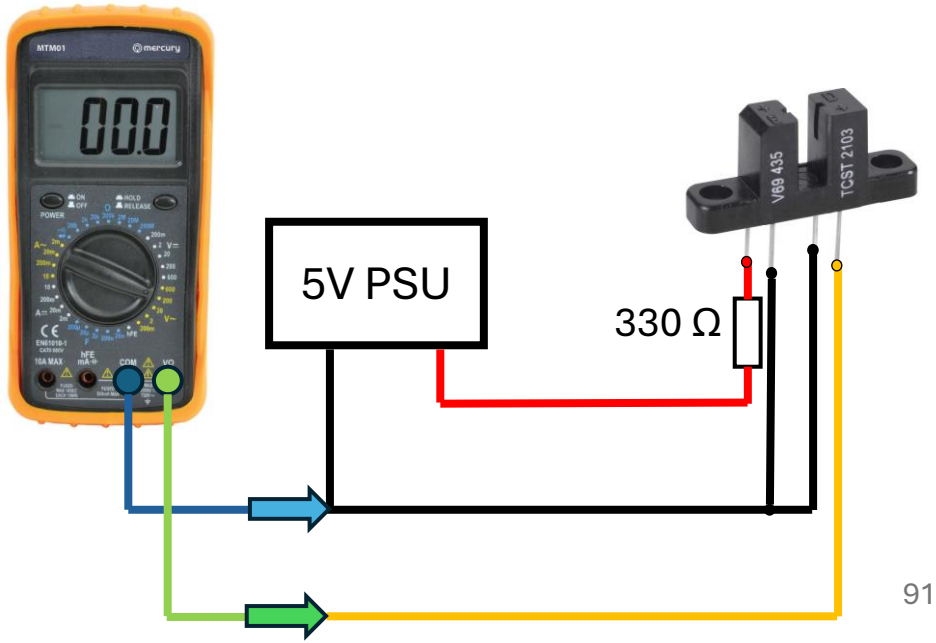
PREPARATION OF ENDSTOPS

Prepare the endstops (by soldering the resistor and wires). Heat shrink tubing has be added to prevent any shorts between the soldered connections.



NOTE

Powering the sensor with an incorrect polarity will likely result in permanent damage. Make sure the wires have been attached correctly before powering the sensor! To check if the sensor has been wired properly, you can connect the sensor to a 5V source and then measure the voltage between GND and S. For a "close" state, the voltage should be 5V and when something enters the gap in the sensor, the state is "open" and the voltage should be close to 0V

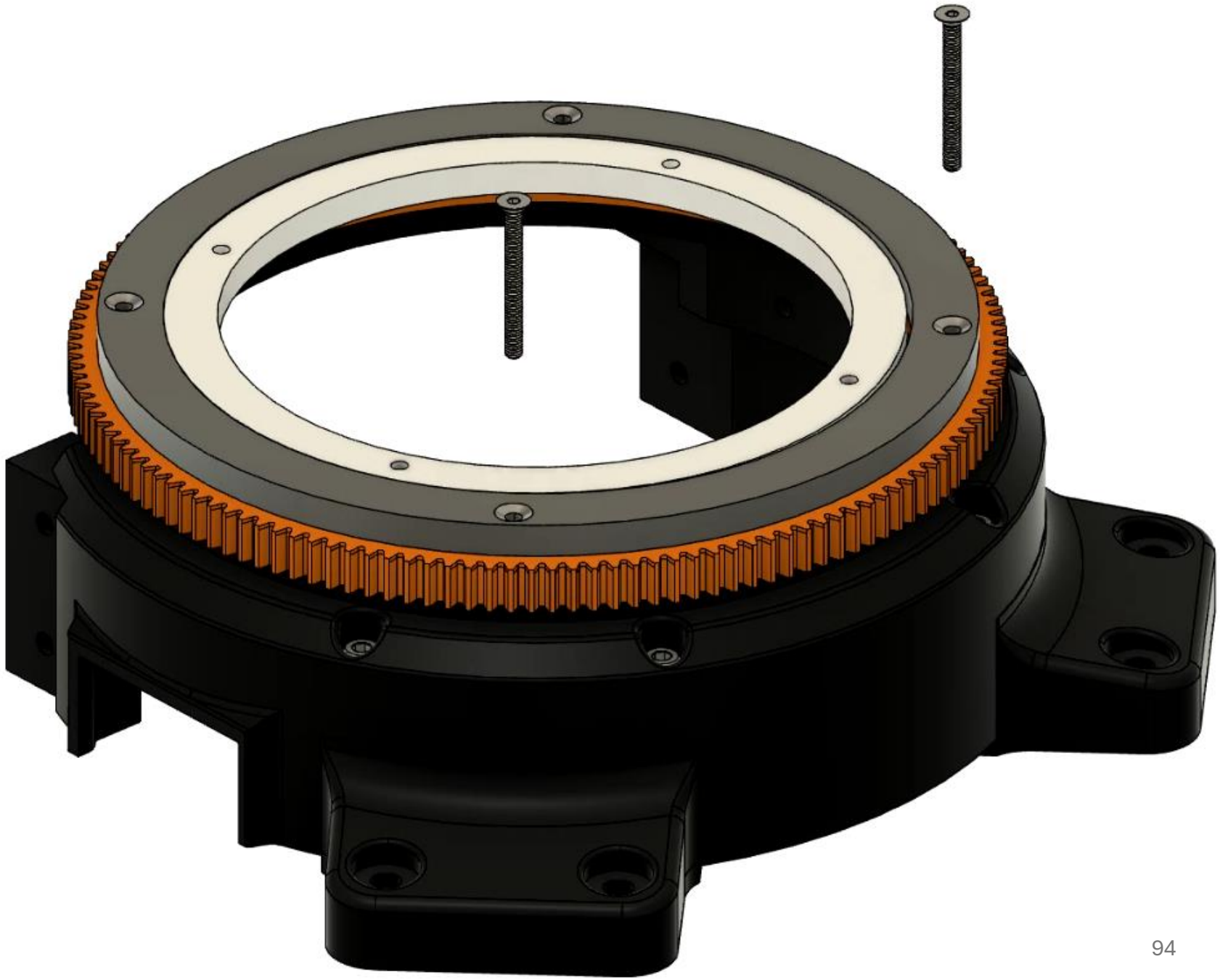


Endstop 1 – wires should be 15-18 cm long
Endstop 2 – wires should be 15-18 cm long
Endstop 3 – wires should be 75-80 cm long
Endstop 4 – wires should be 25-28 cm long
Endstop 5 – wires should be 25-28 cm long
Endstop 6 – wires should be 70-73 cm long

ASSEMBLY

4x M5x40
4x M5 nut

2x M4x40
2x M4 nut

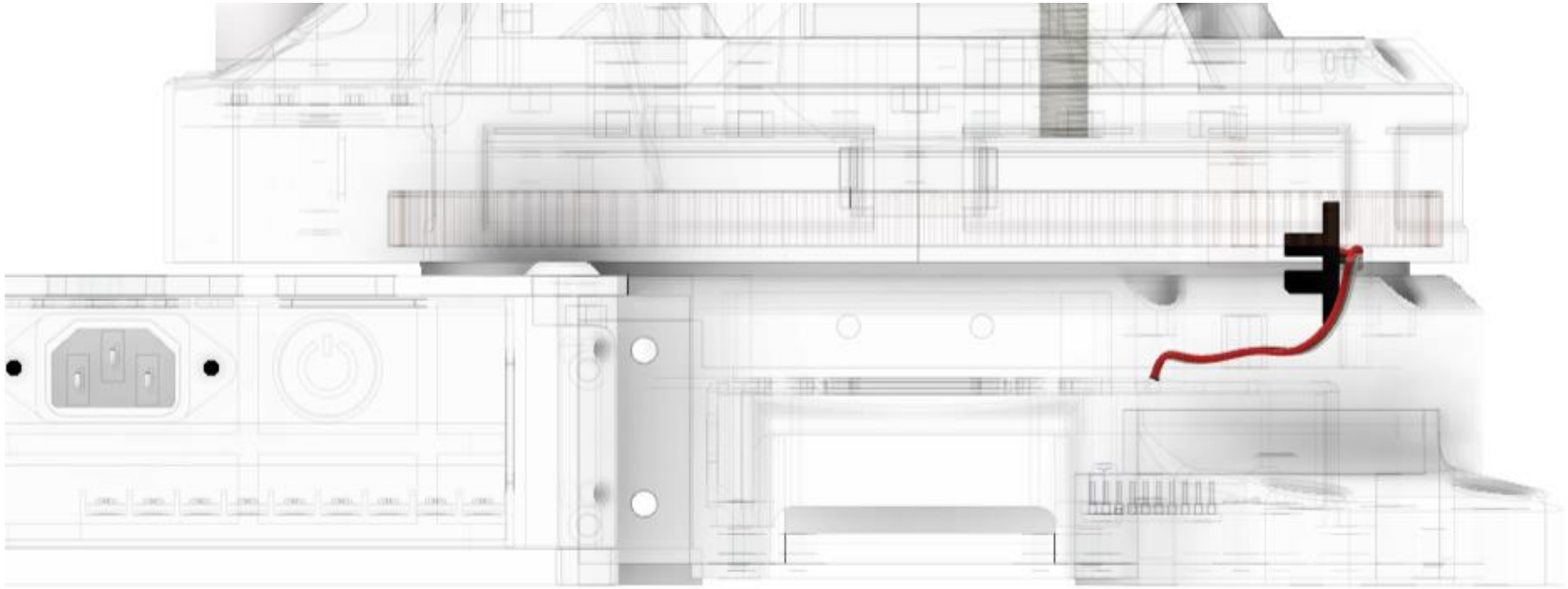


Note:
Don’t insert all 4 screws!

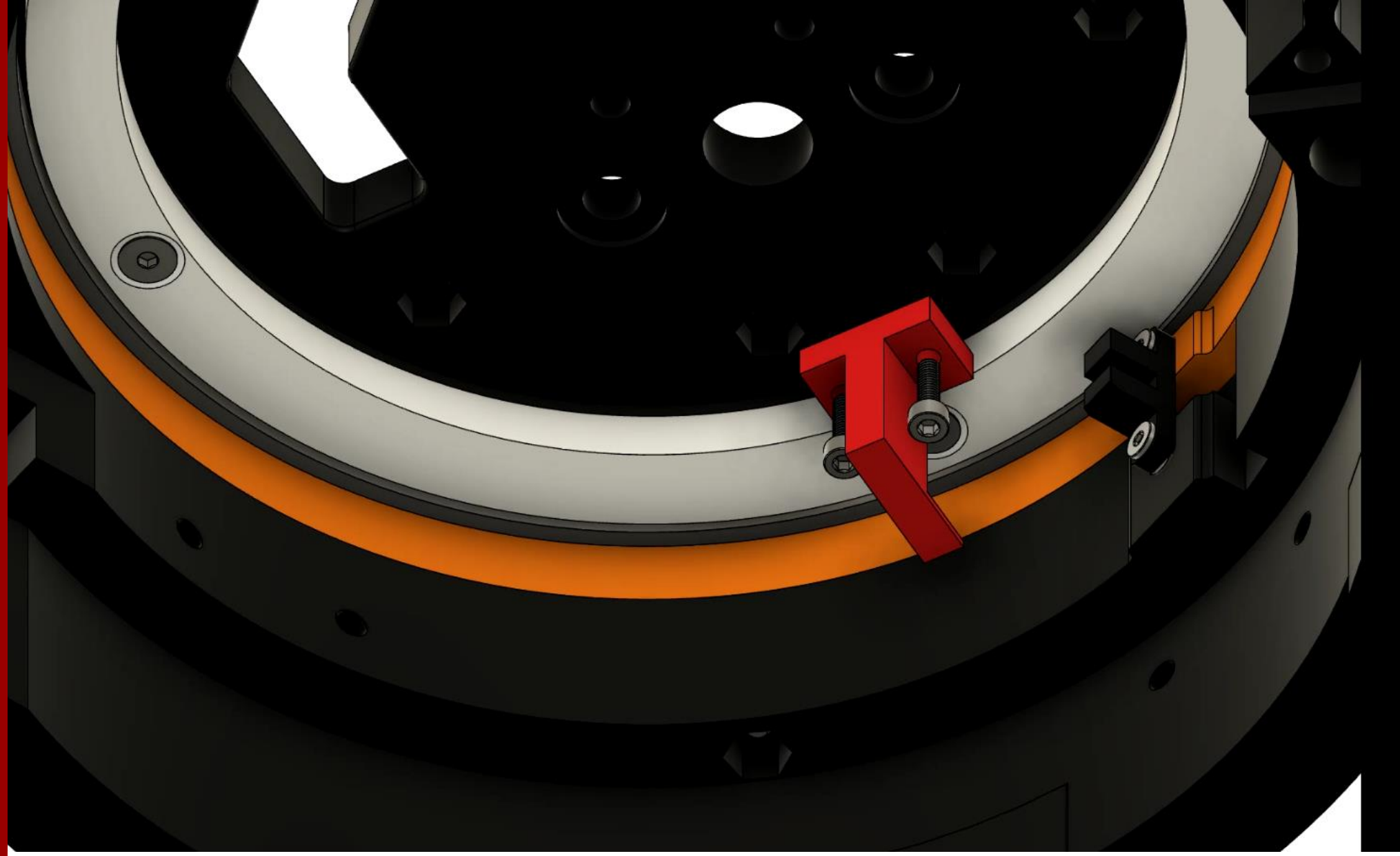
2x M3x8
1x TCST2103

4x M4x20
4x M4 nut

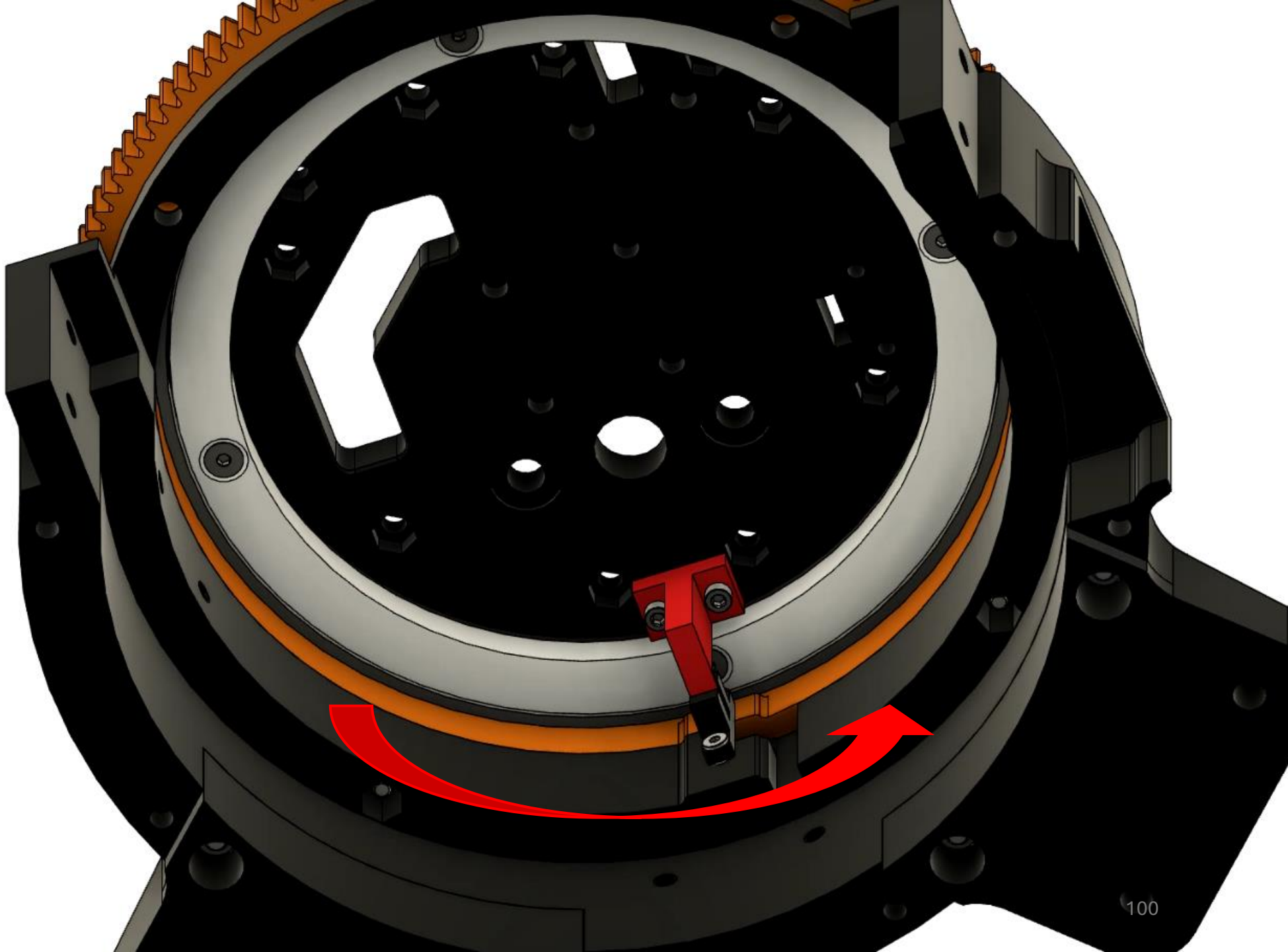
SENSOR WIRING
Do not connect the sensor To the motherboard yet. The electronics will be connected after the assembly

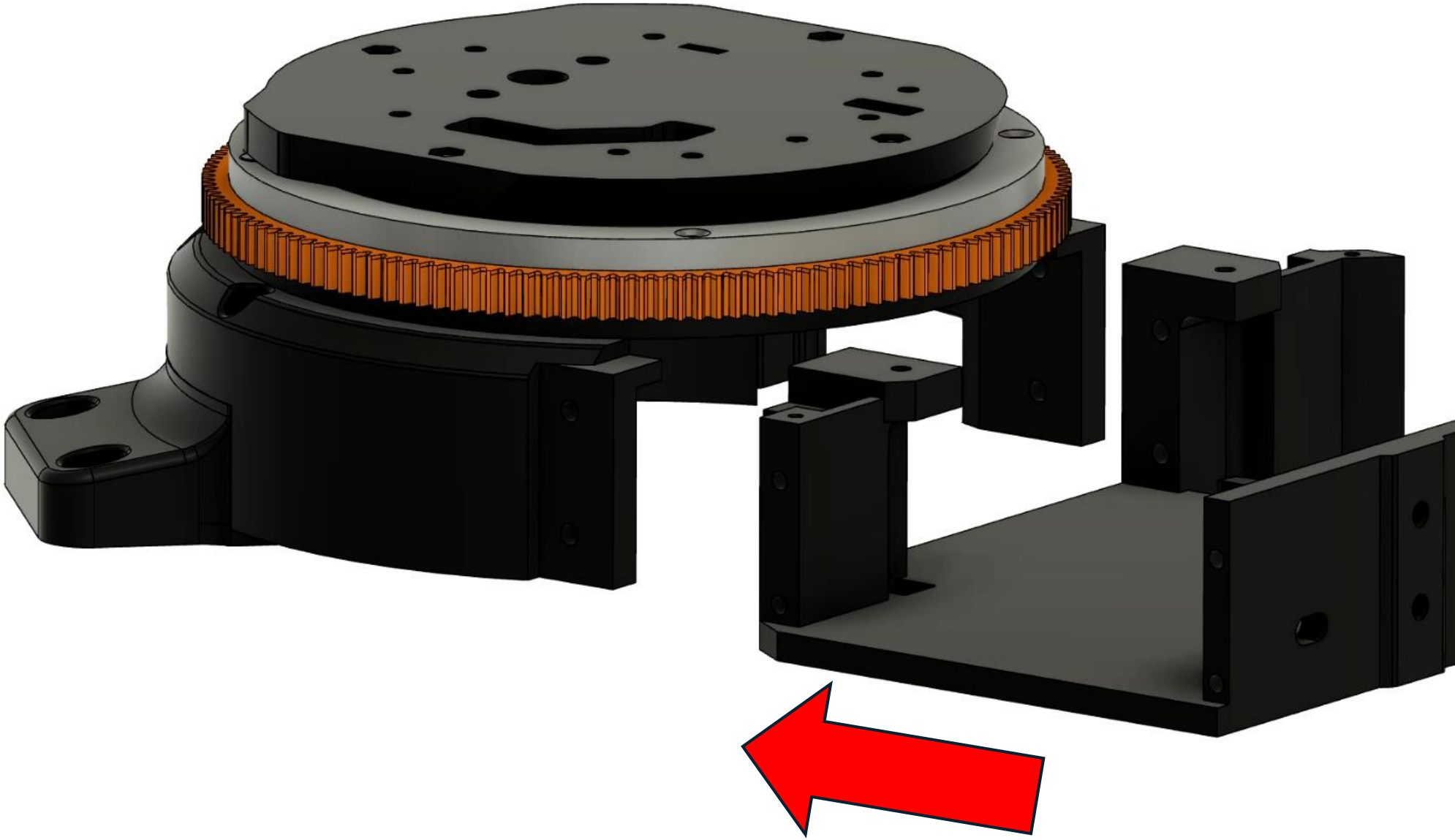


2x M3x10



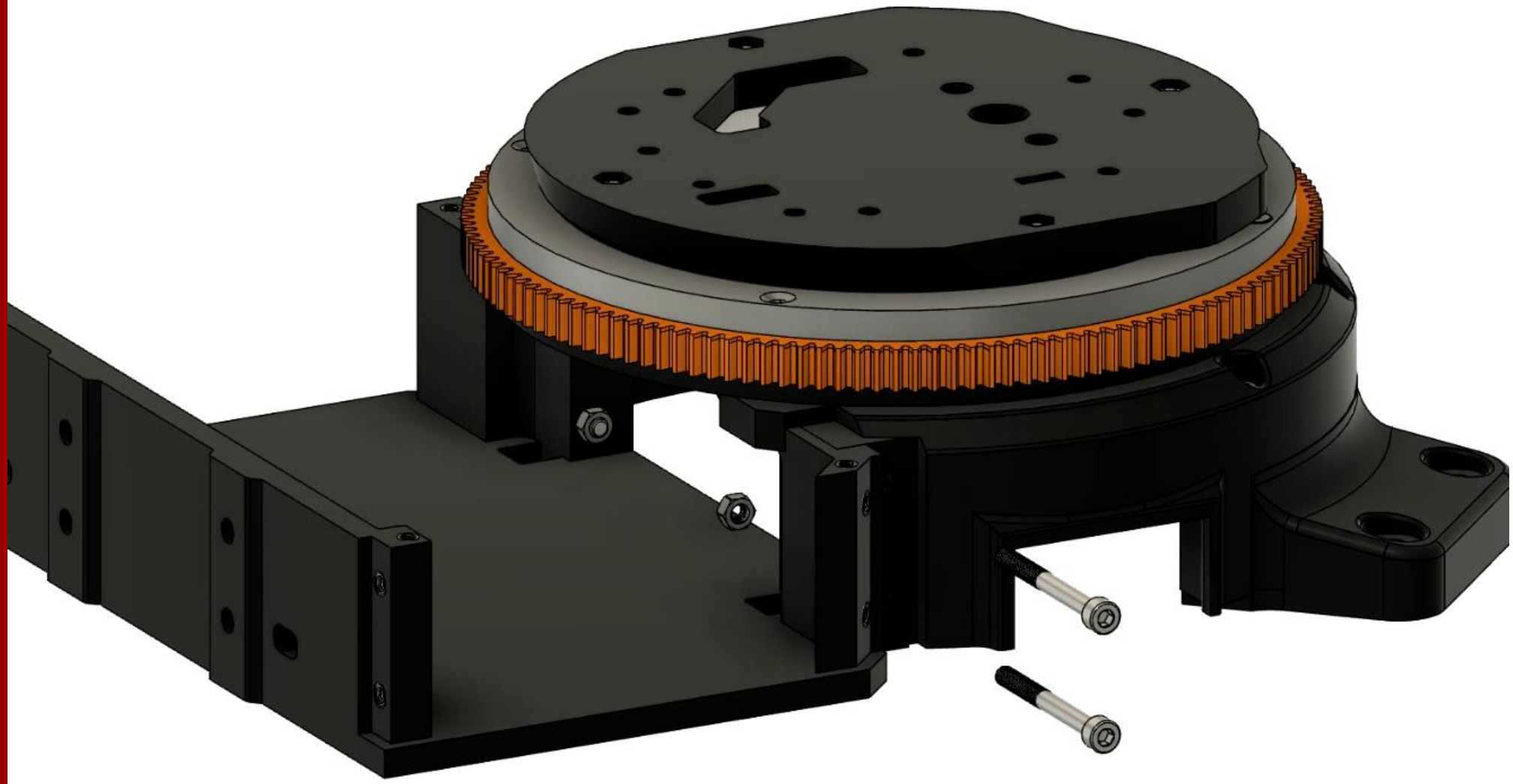
Check if sensor trigger fits in the sensor gap by rotating the platform





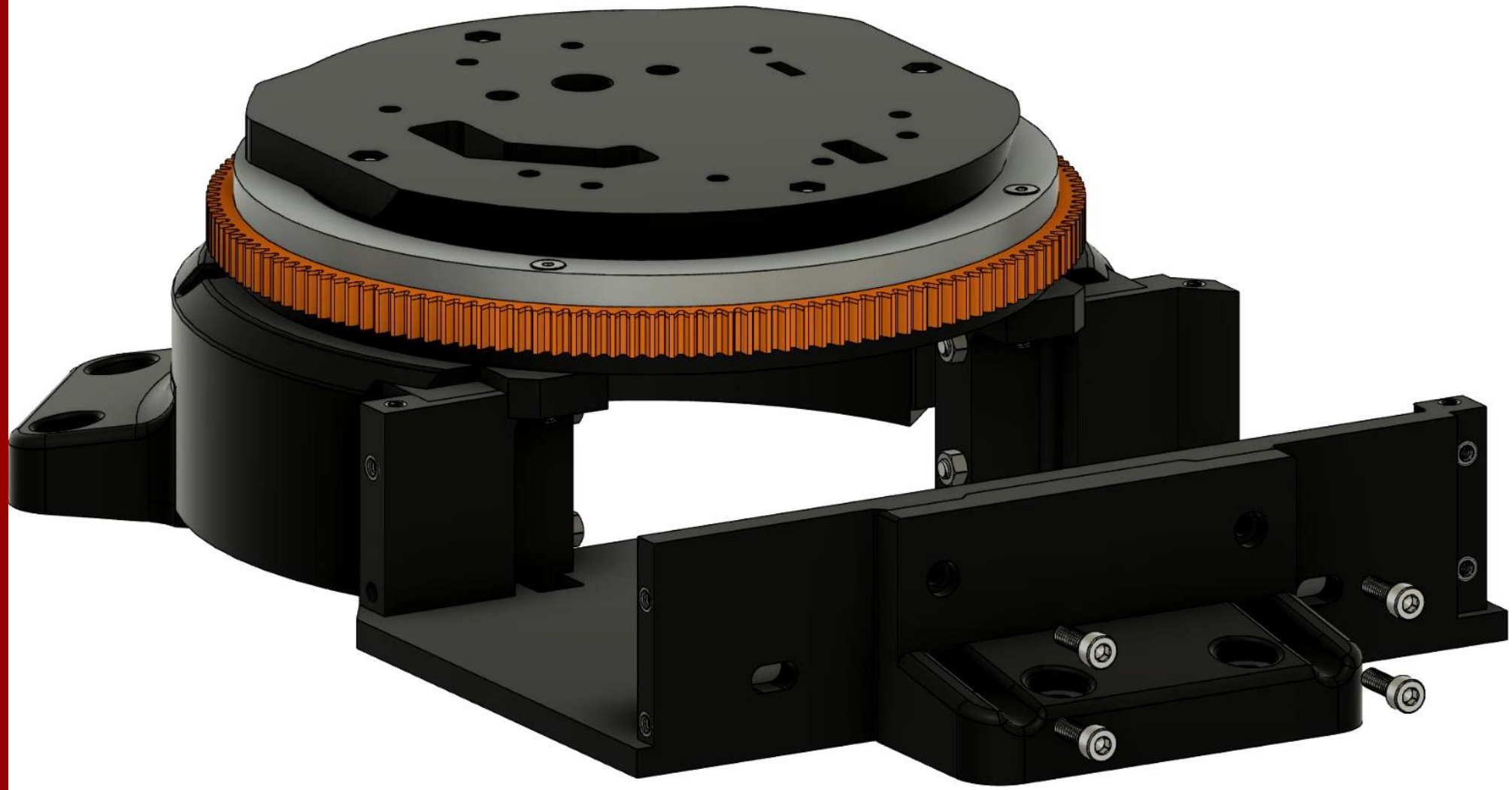
2x M5x50
2x M5 nut

2x M5x50
2x M5 nut

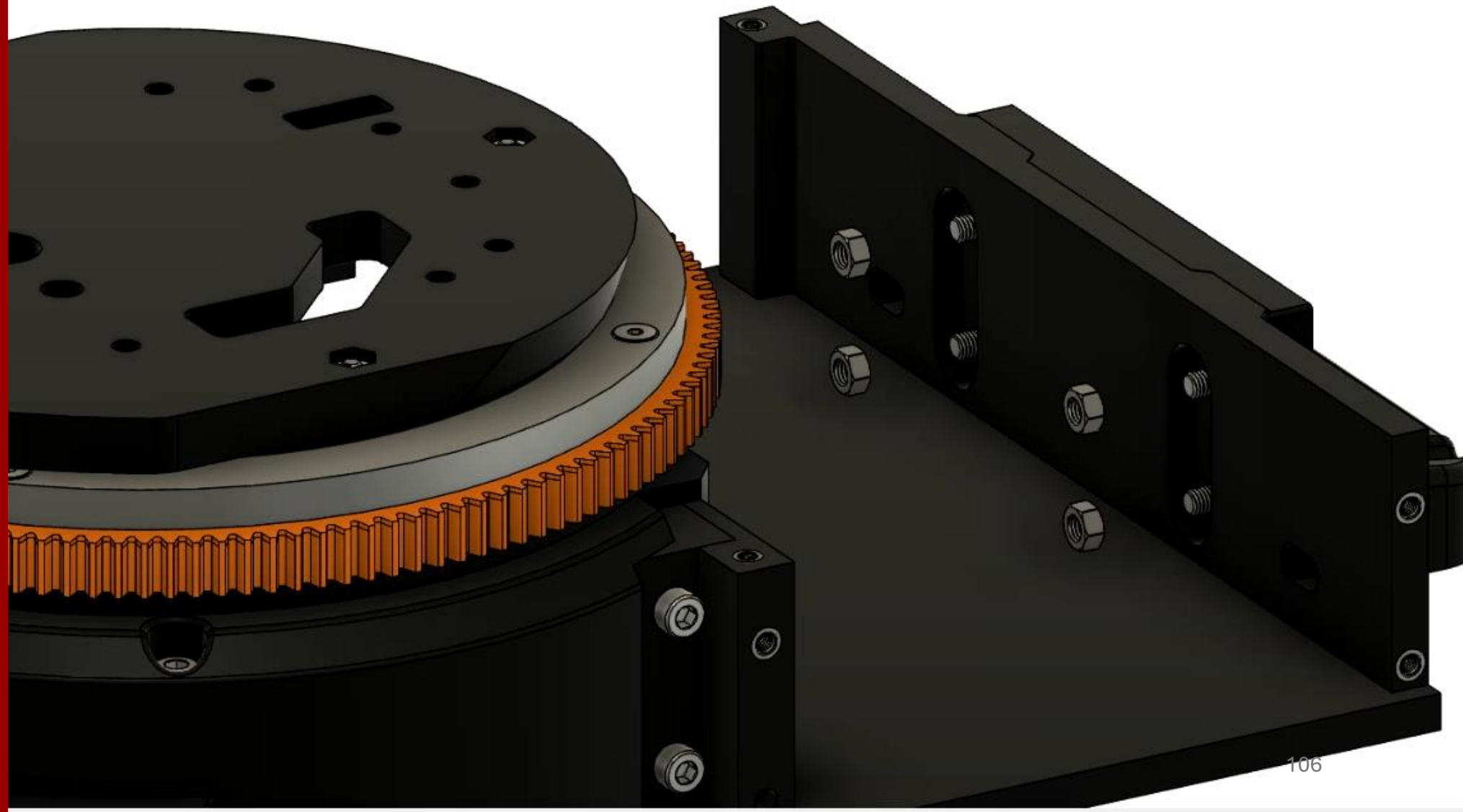


2x M4x40
2x M4 nut

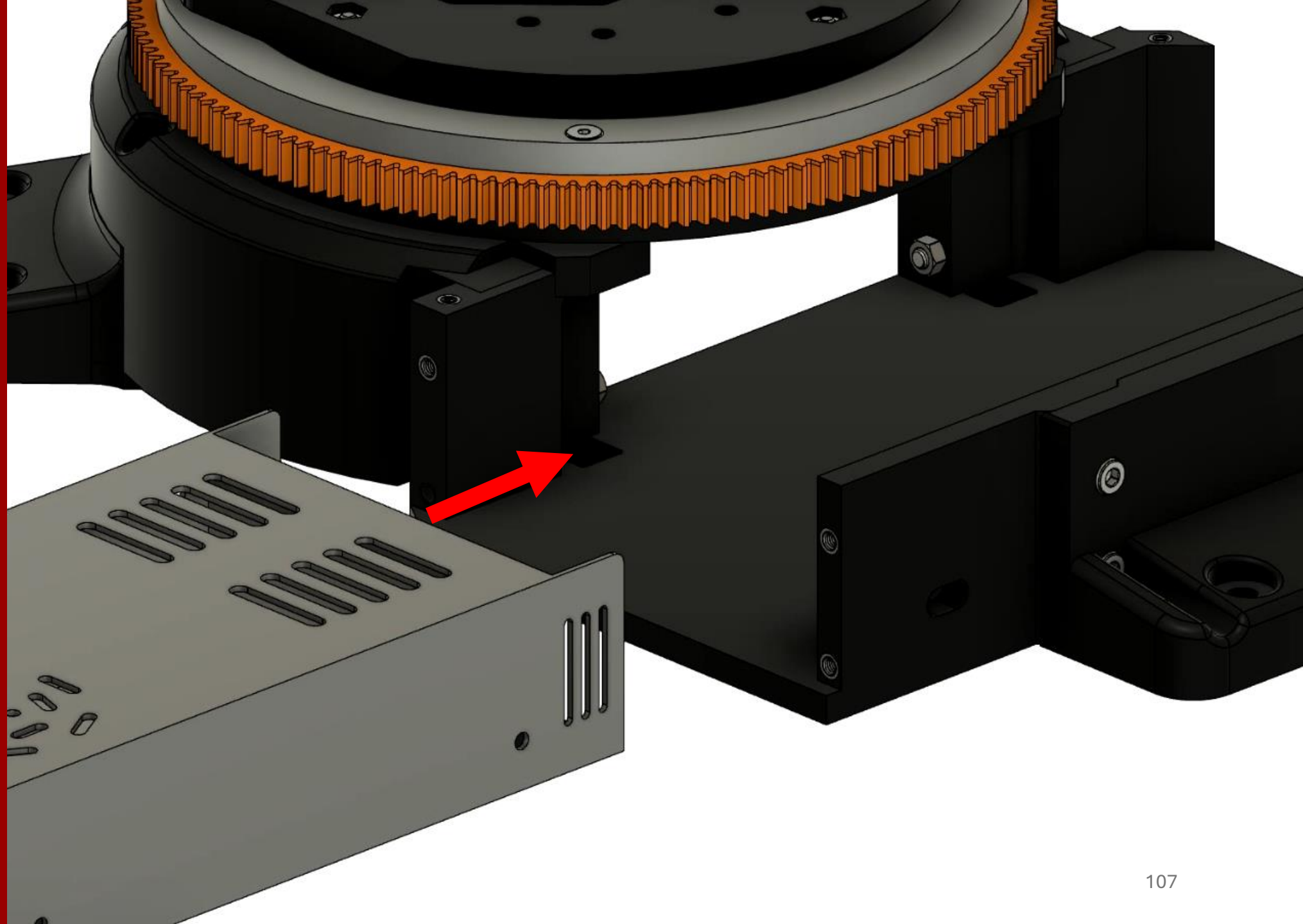
4x M5x15



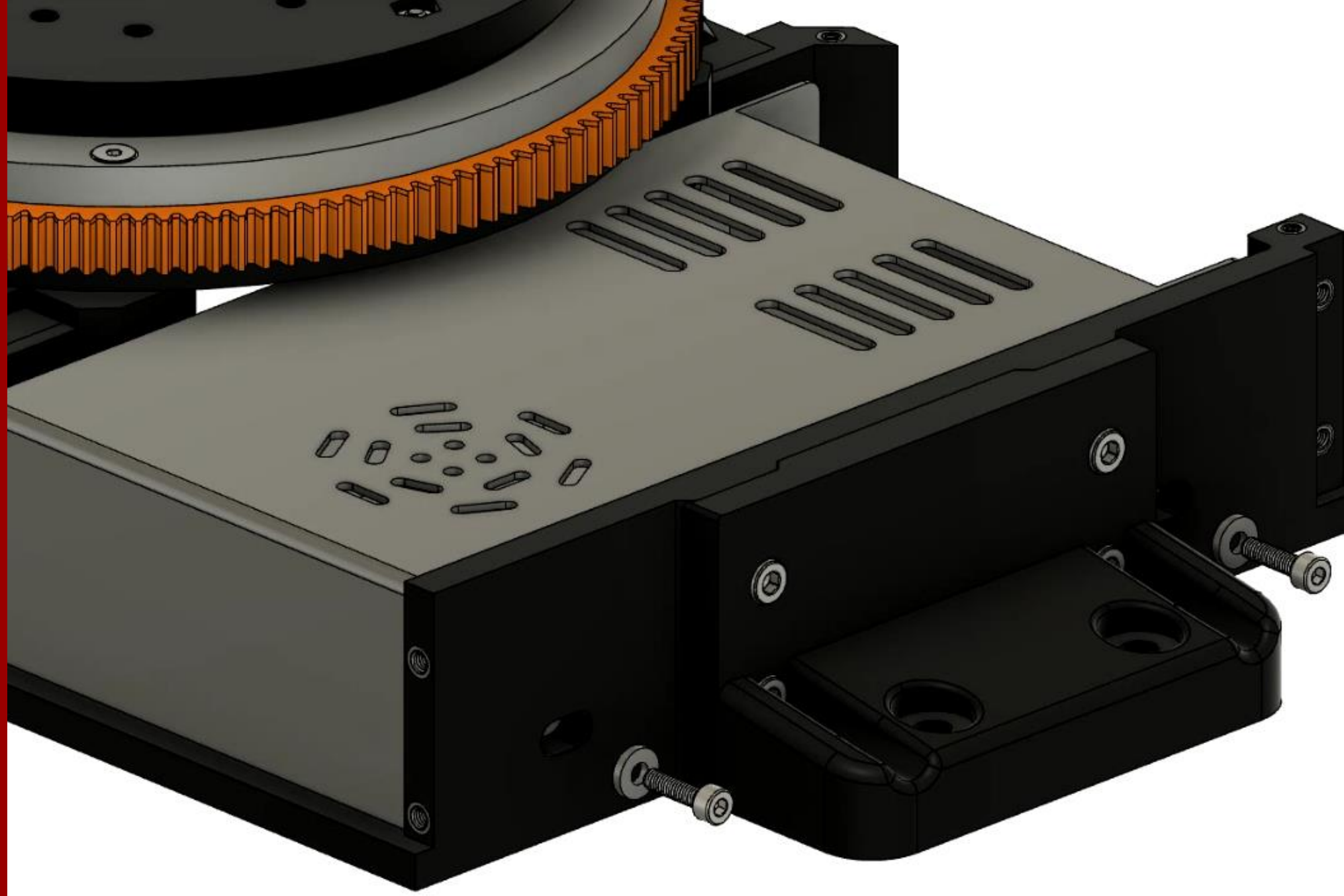
4x M5 nut



Insert power supply into the PSU box

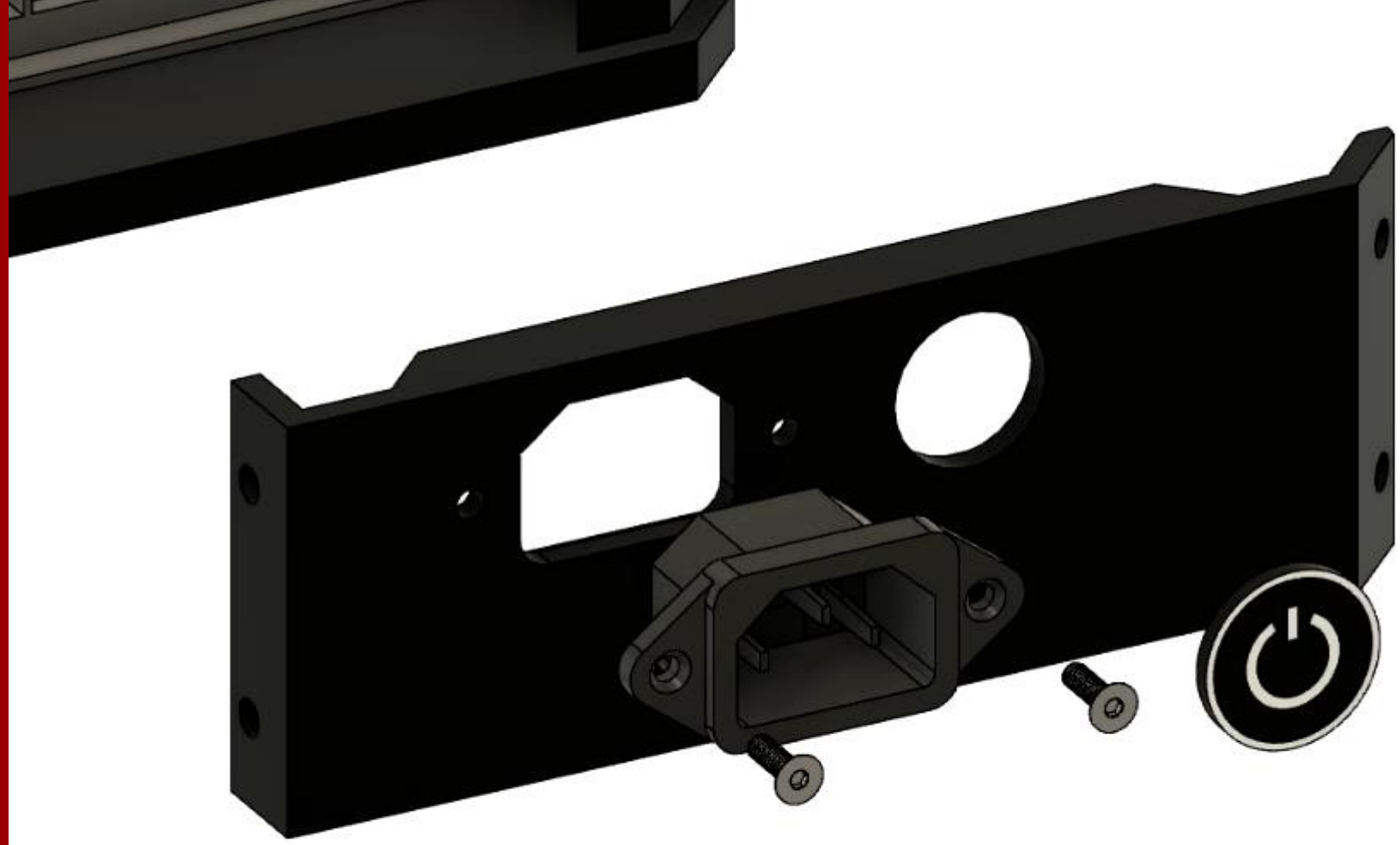


2x M4x15
2x M4 washer



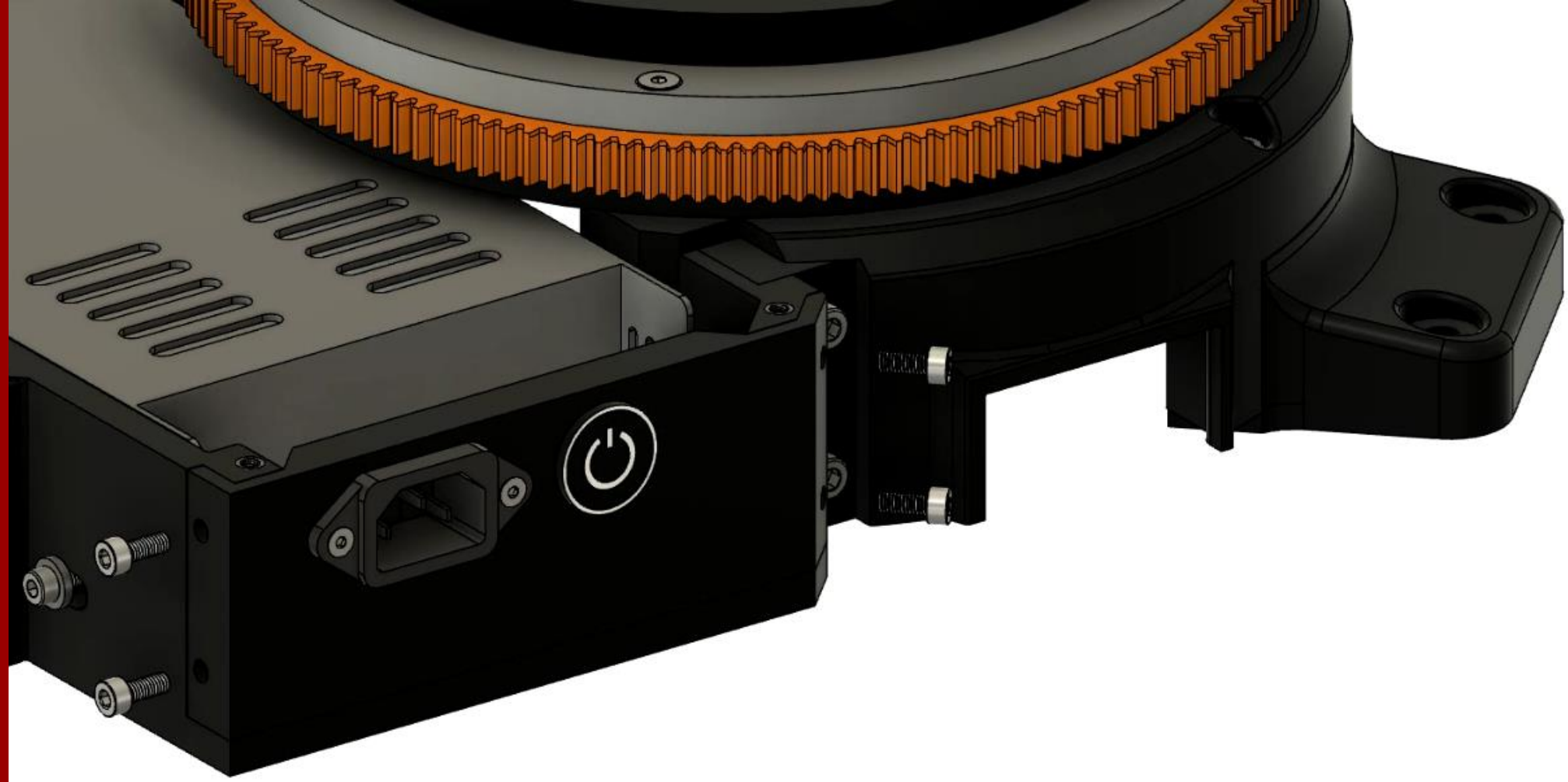
4x M4x10

2x M3x10 flat head screw
1x Power button
1x C14 socket

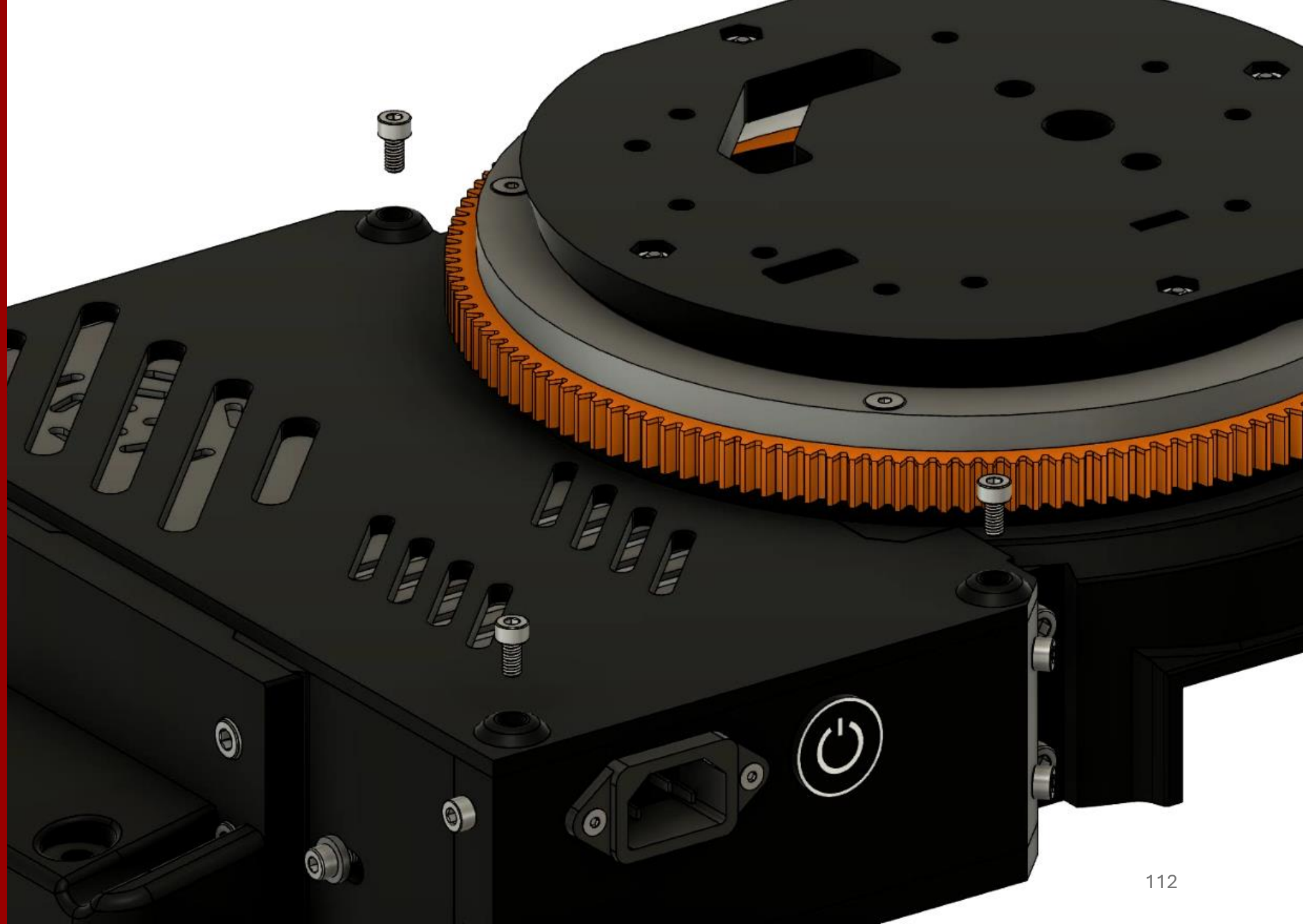


The front panel button and socket wiring have been described later in the manual

4x M4x10



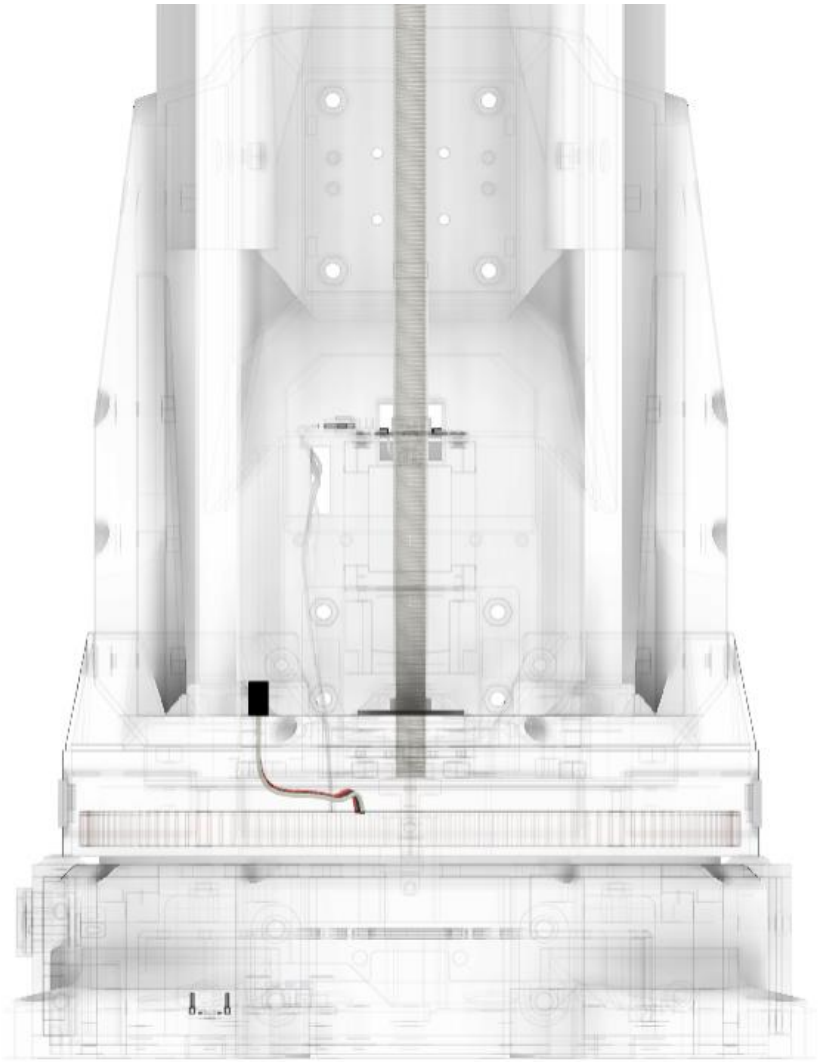
3x M4x8

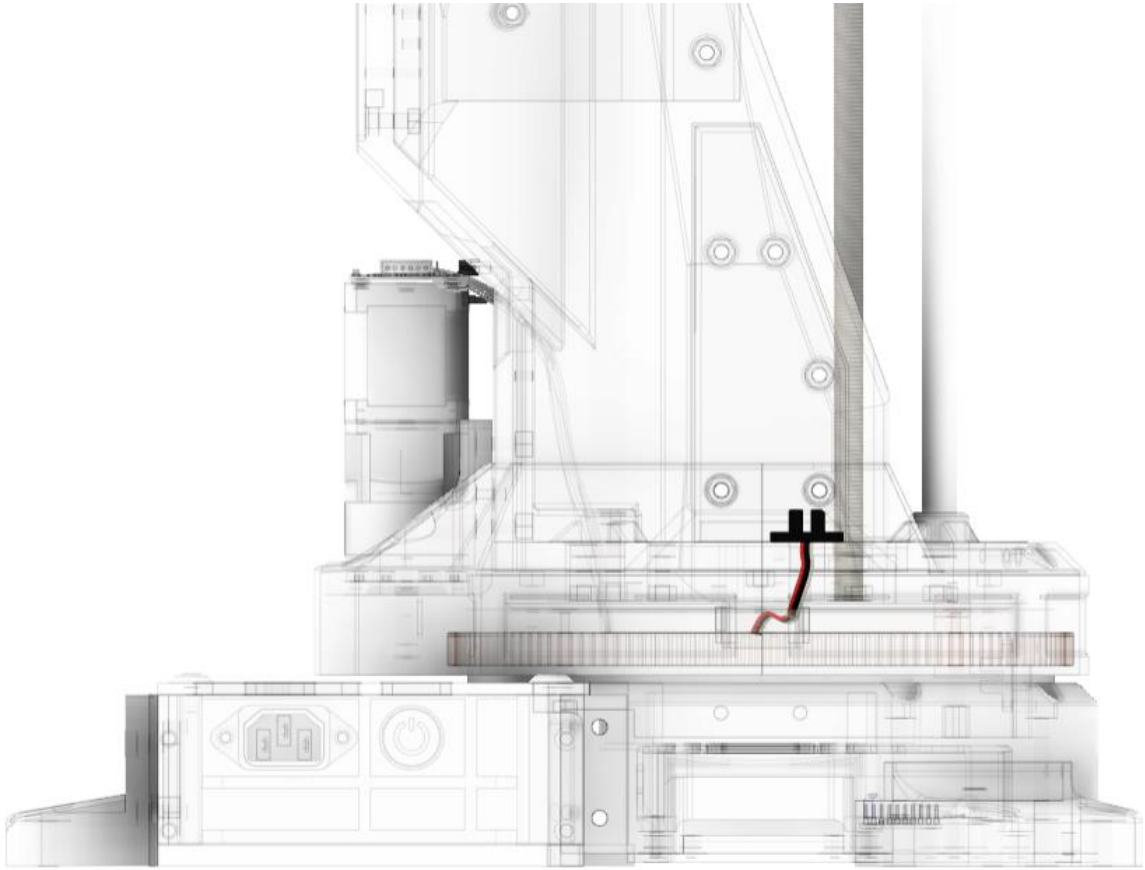


1x 10mm Pillow ball bearing
10x M5x20
10x M5 nut

2x M5x30
2x M5 nut

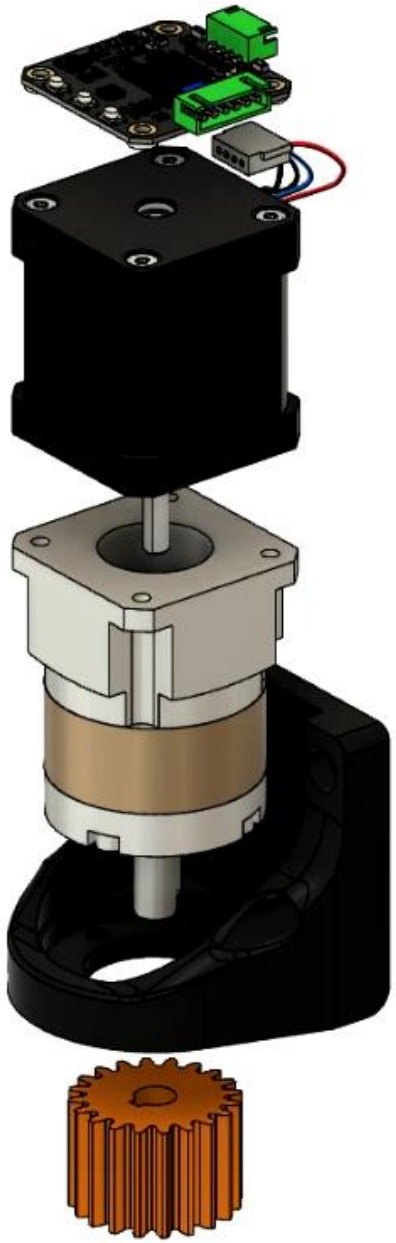
2x M3x8
1x TCST2103





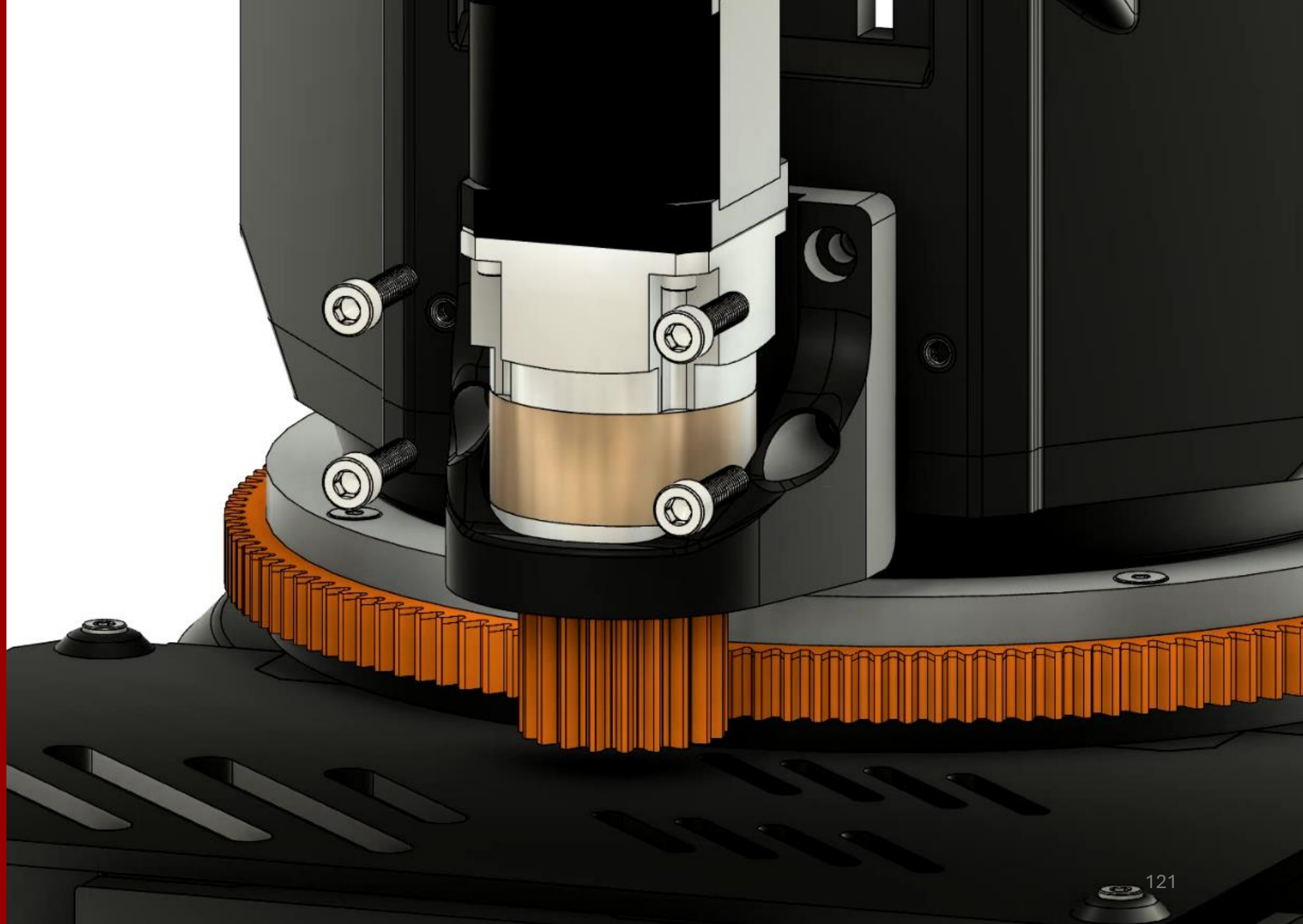
6x M5x20
6x M5 nut

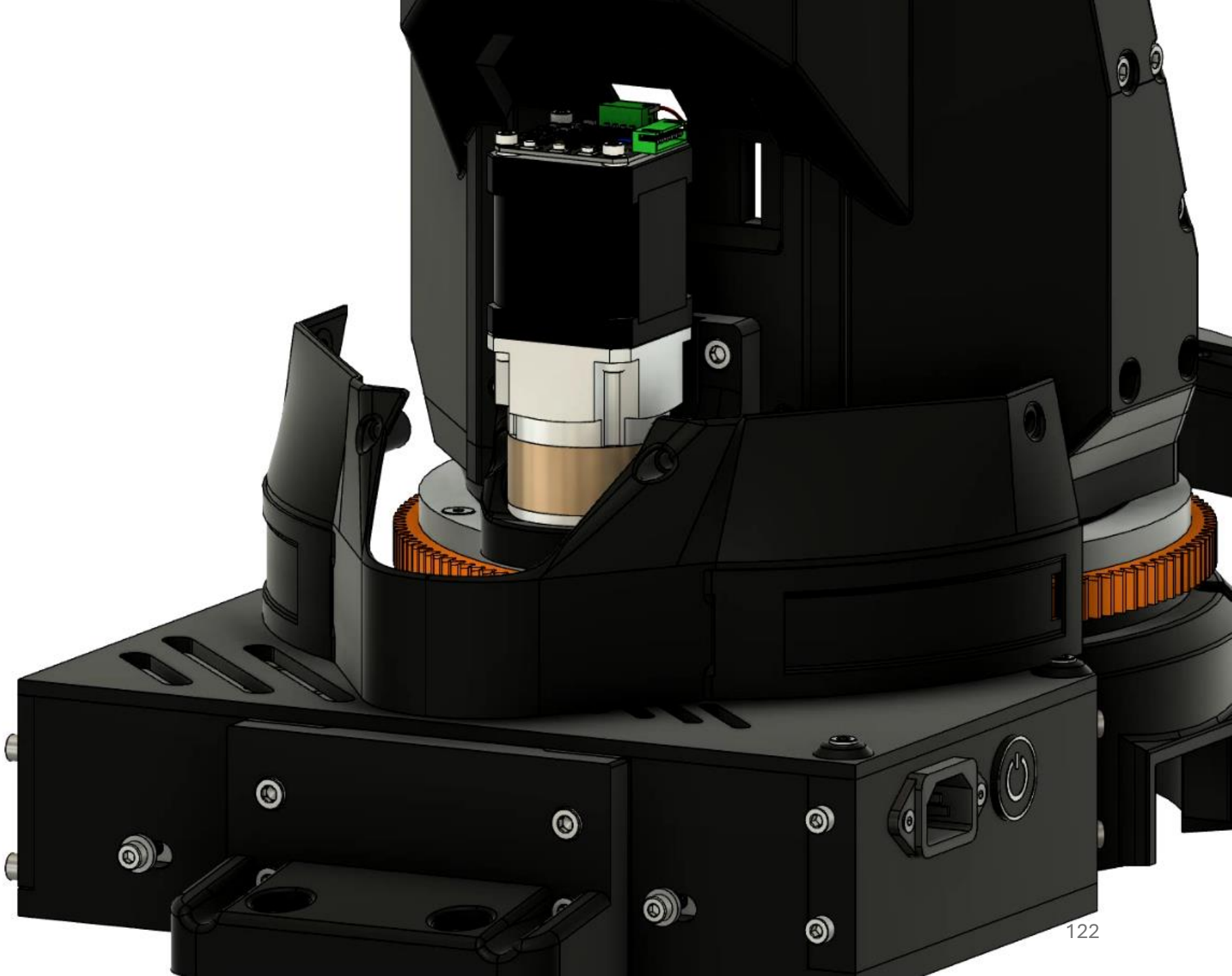
1x MKS Servo42C
1x Nema 17 (17HS19-2004S1)
1x Planetary Gearbox 5:1
4x Distance washer



4x M3x8
4x M3x10
4x M4x8 flat head screw
1x M3x10

4x M5x20
4x M5 nut



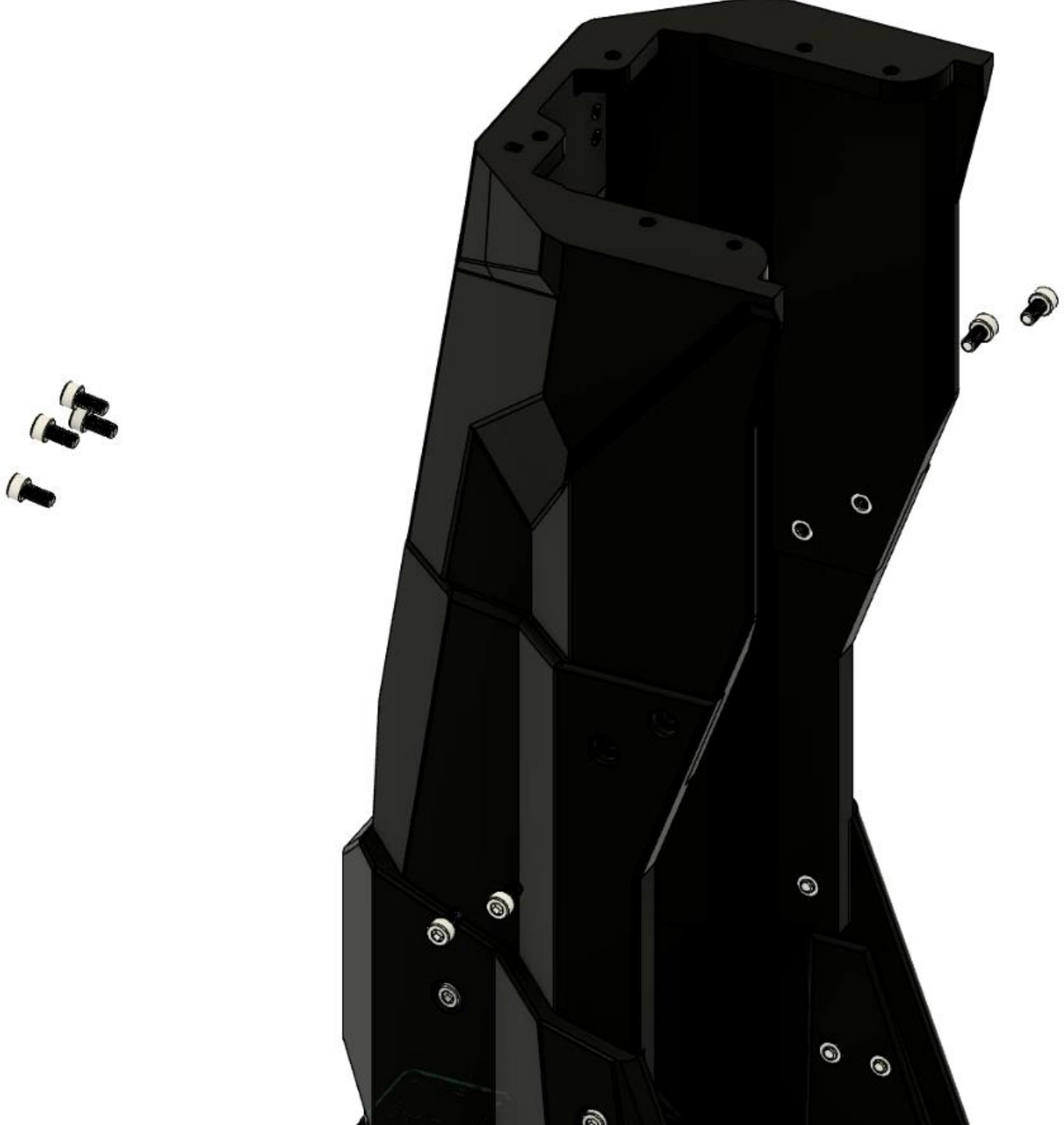


2x M4x10

4x M5x30
2x M4x10
4x M5 nut

2x M5x20
2x M5x10
4x M5x15
8x M5 nut

8x M5x10
8x M5 nut

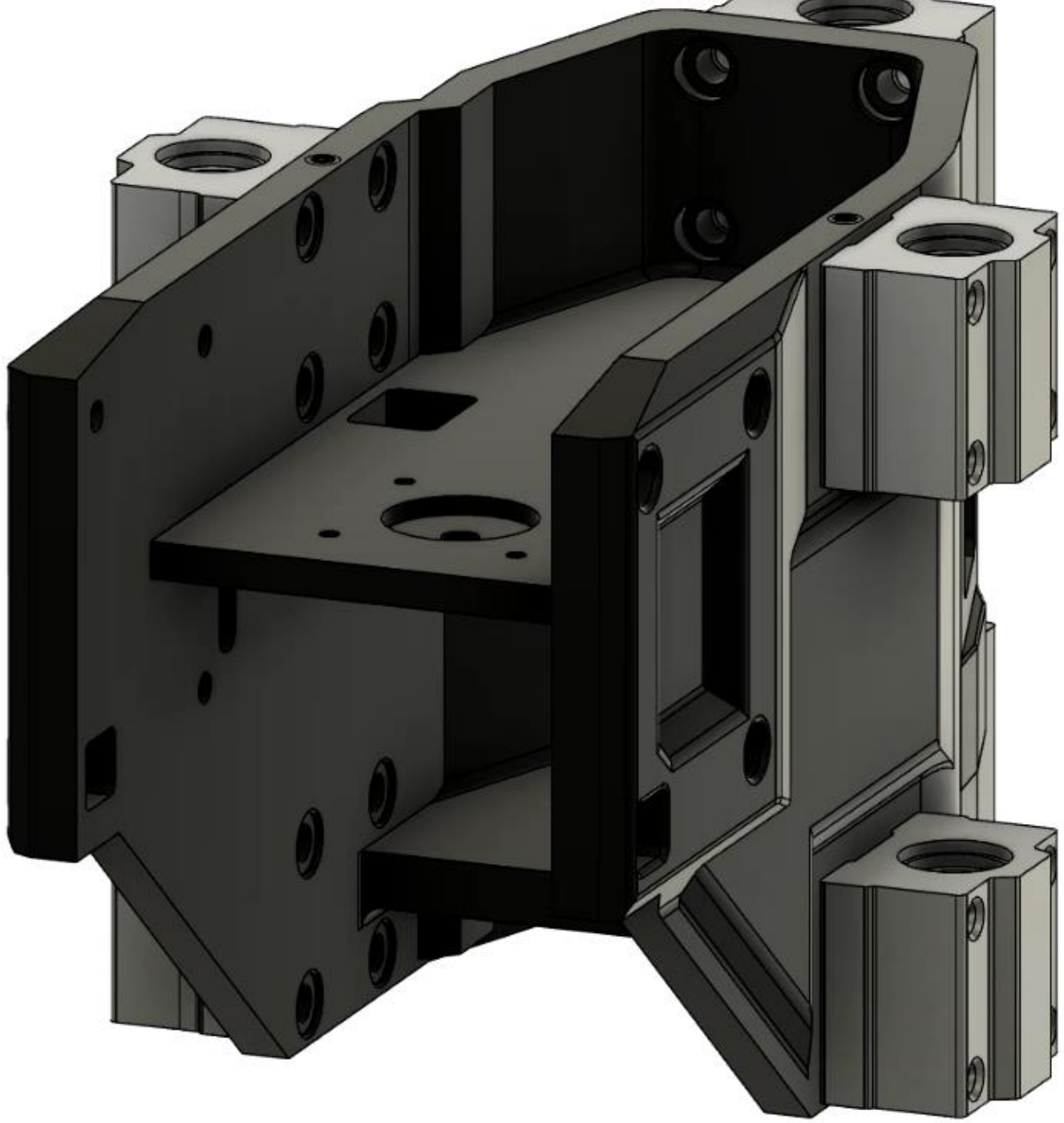



DTU

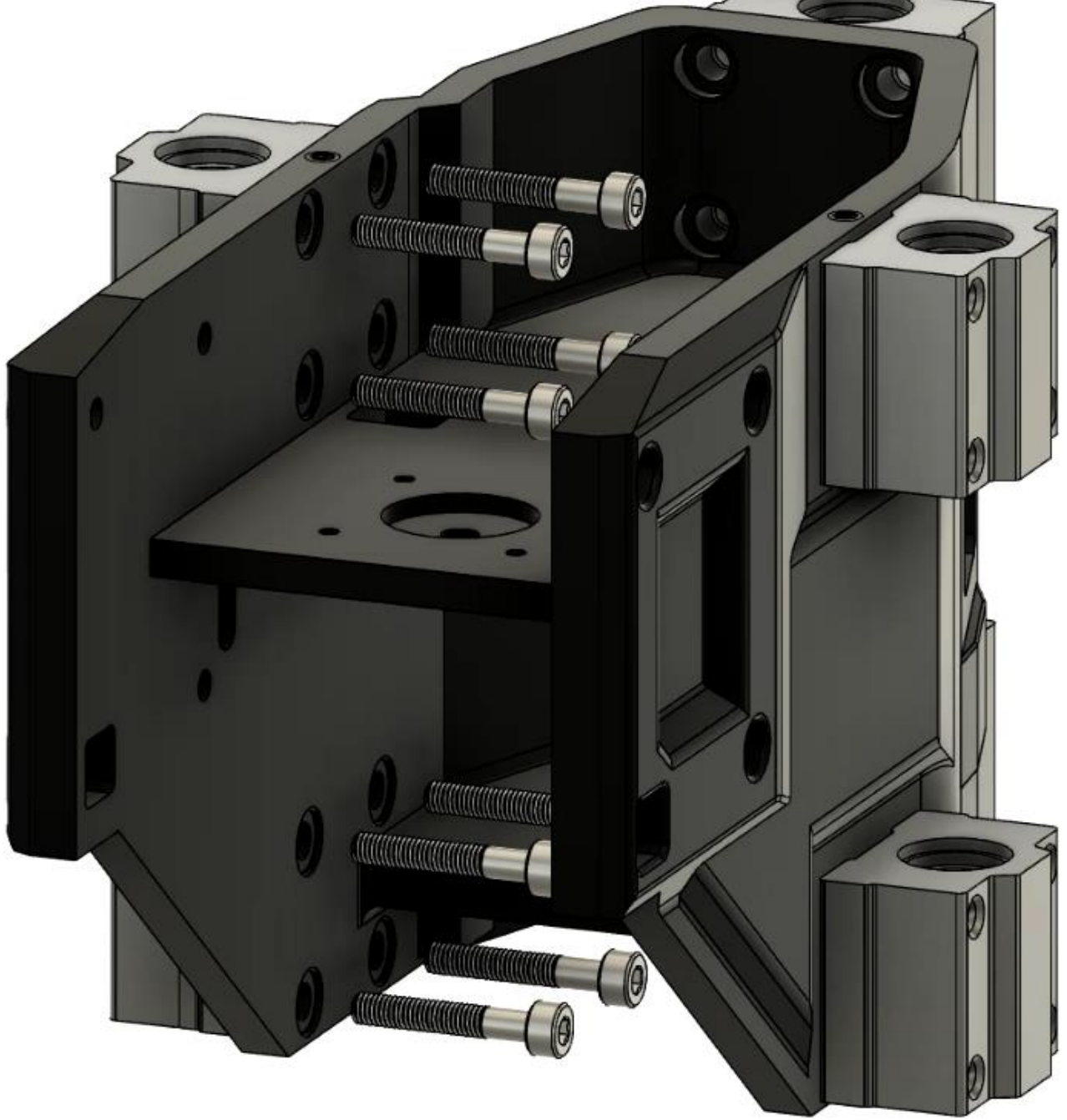
4x M3x15

3x Steel rod 500x12
1x 4mm Lead screw 500x10

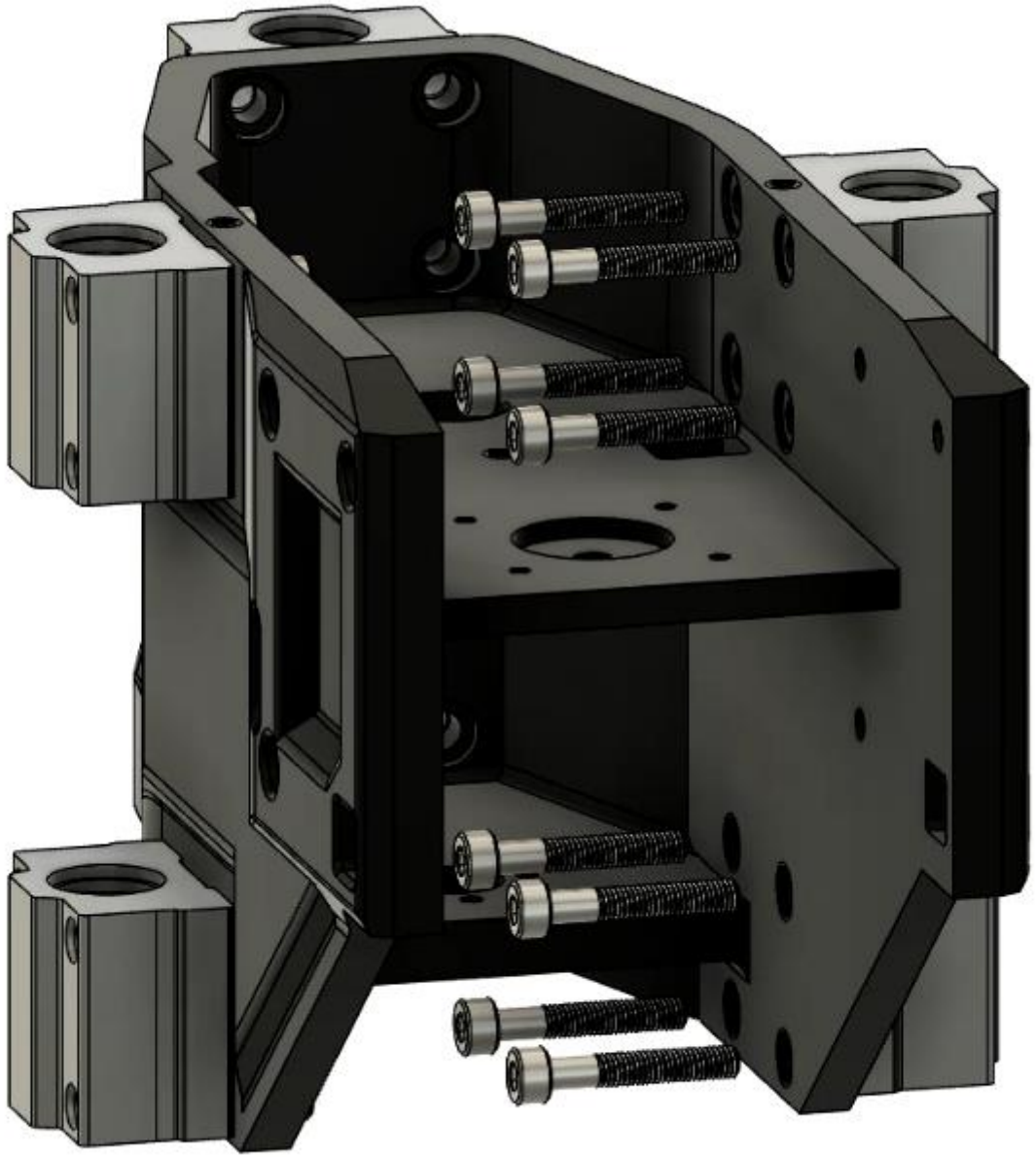
6x Linear ball bearing block



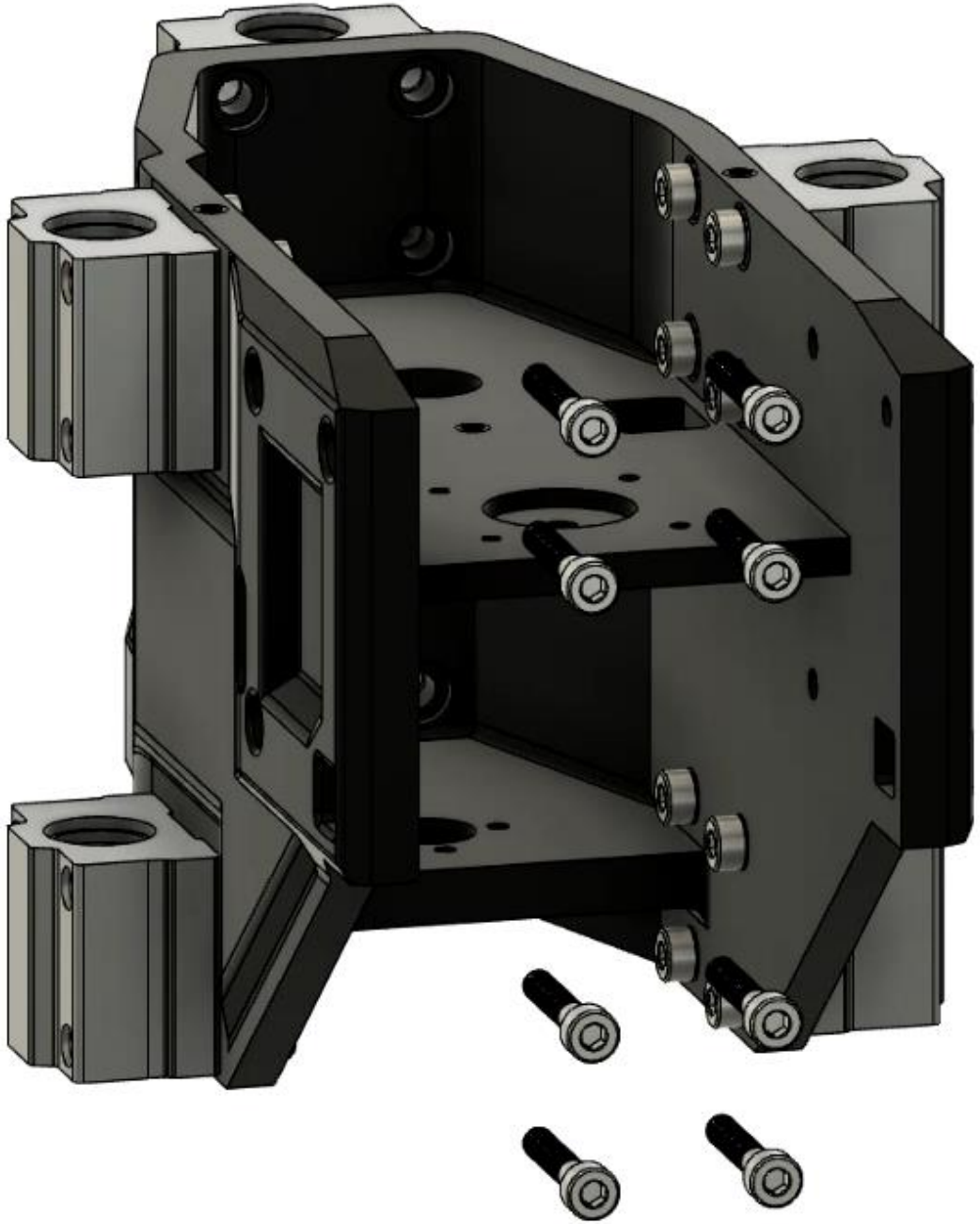
8x M5x15



8x M5x15

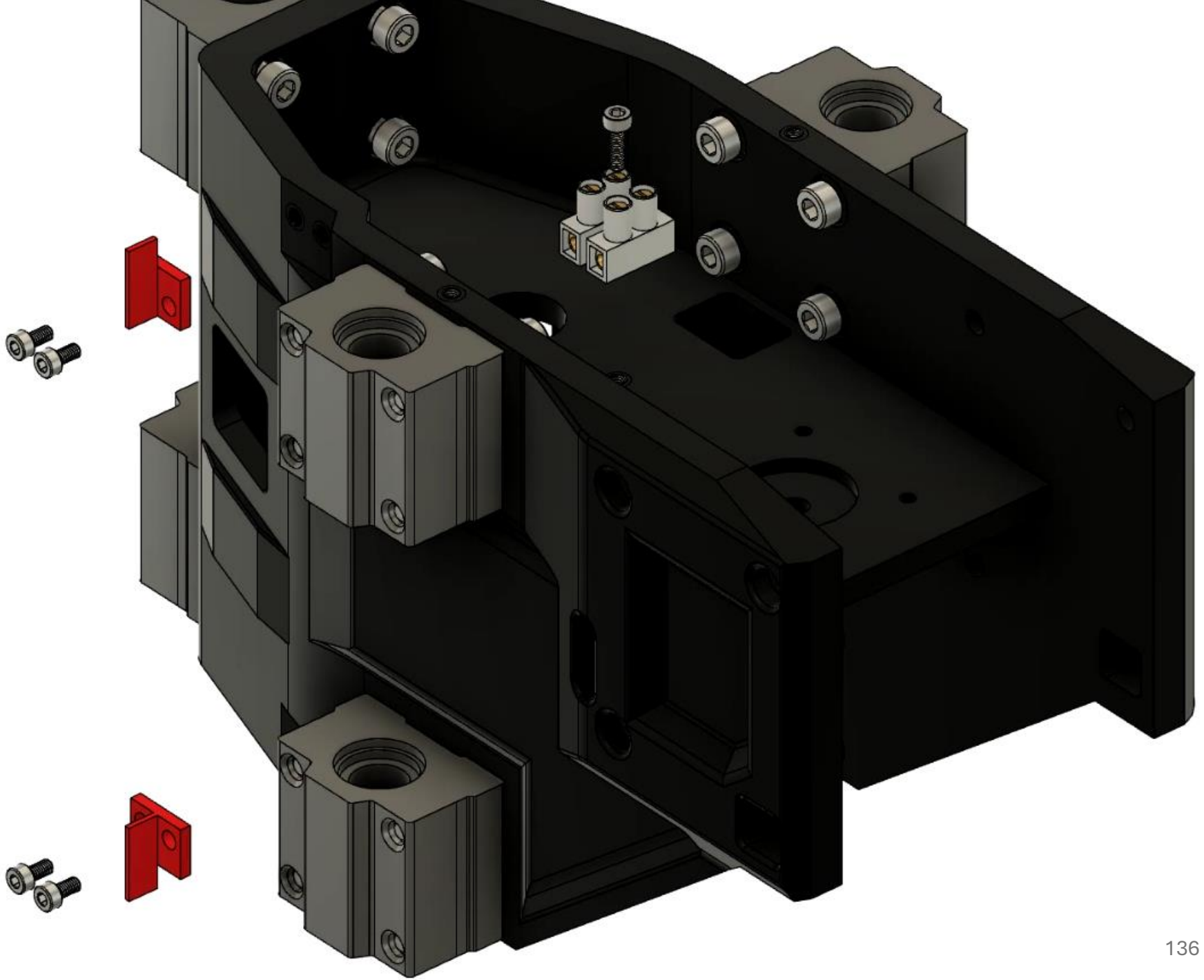


8x M5x15

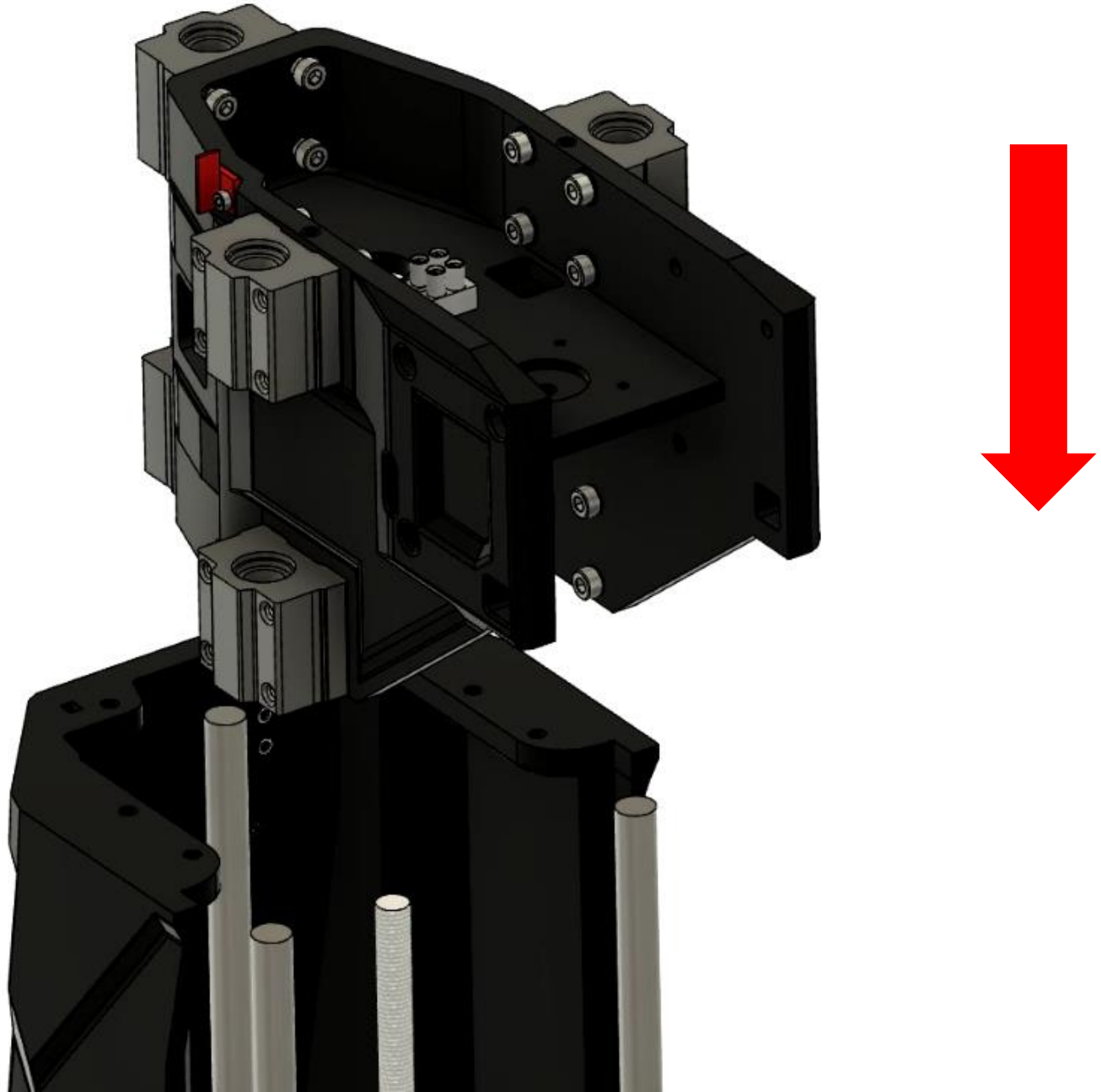


1x Lead screw nut
4x M3x20
4x M3 nut

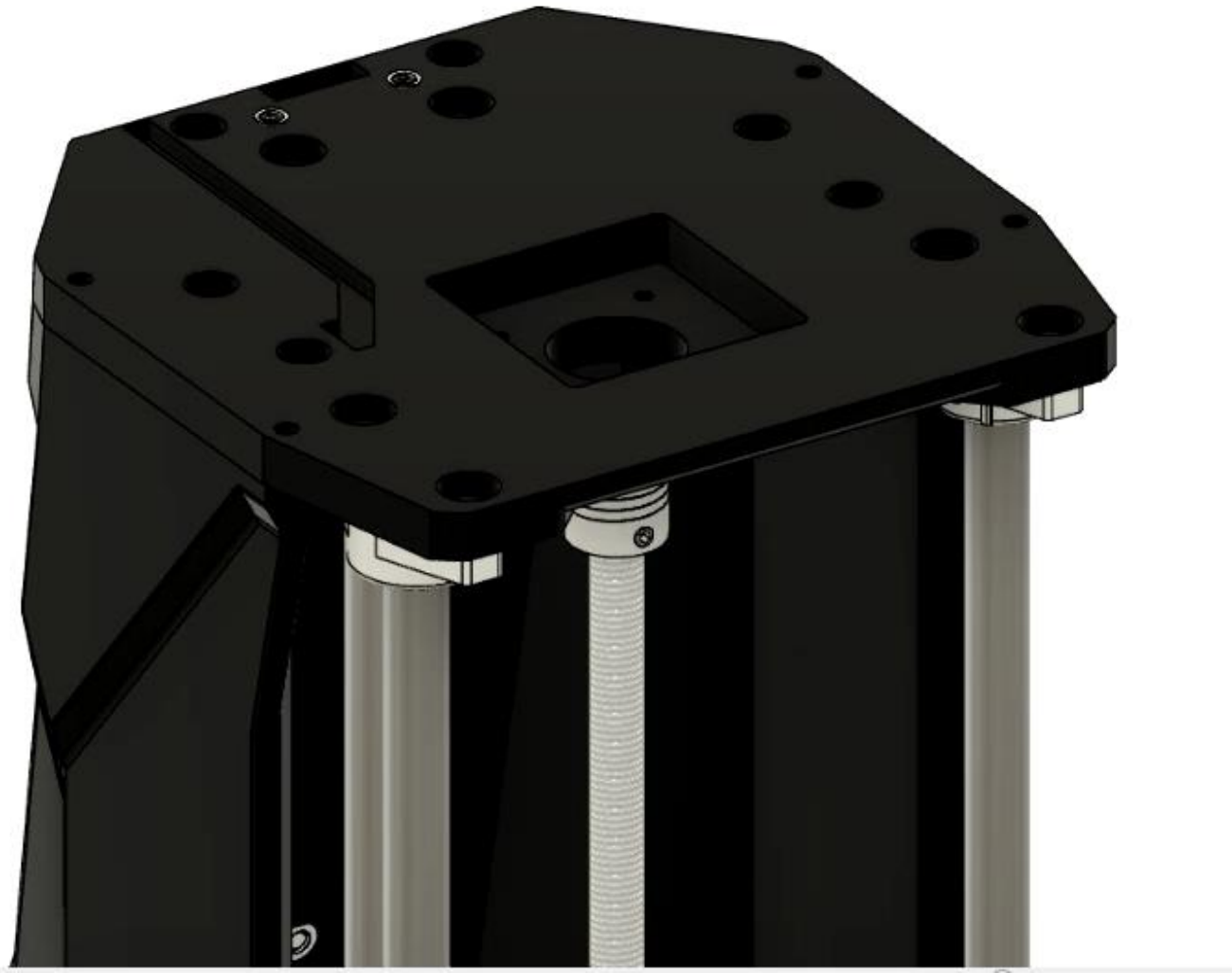
1x Dual row terminal strip
4x M3x8
1x M3x10



Slide arm onto rods and check if lead screw rotates freely.

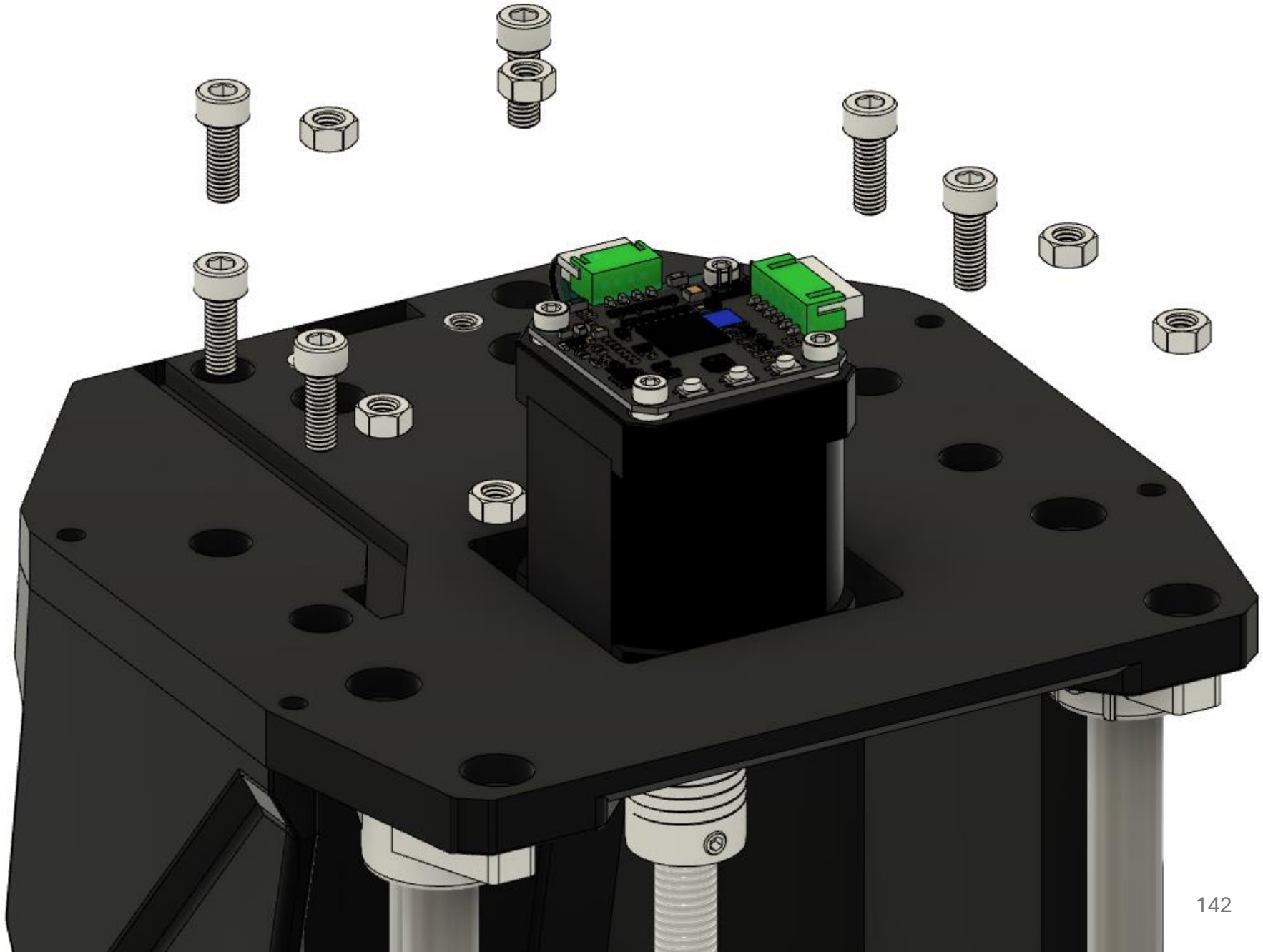


1x Shaft coupling 5x10
3x Rod support SHF12

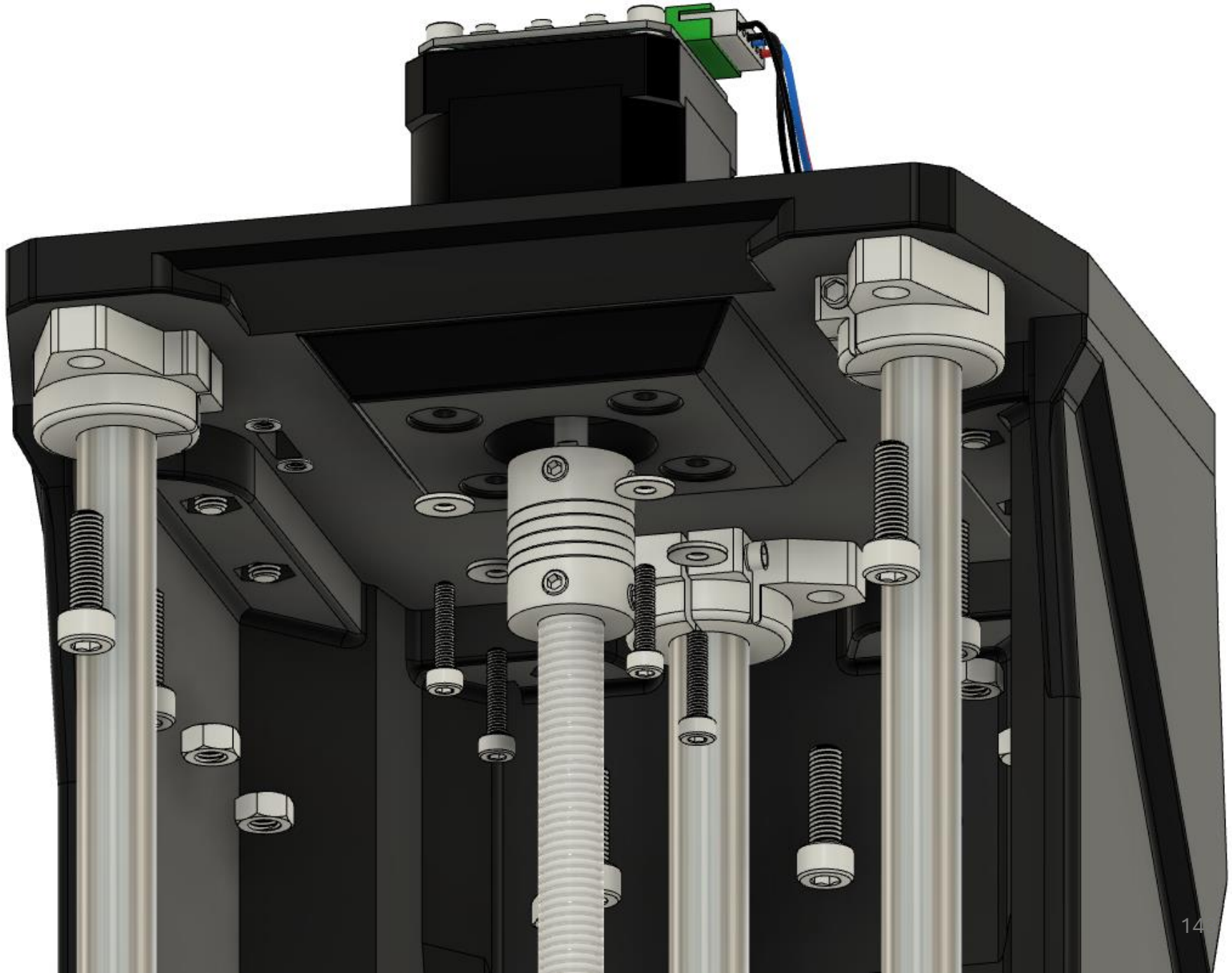


1x Nema 17 (17HS19-2004S1)
1x MKS Servo42C

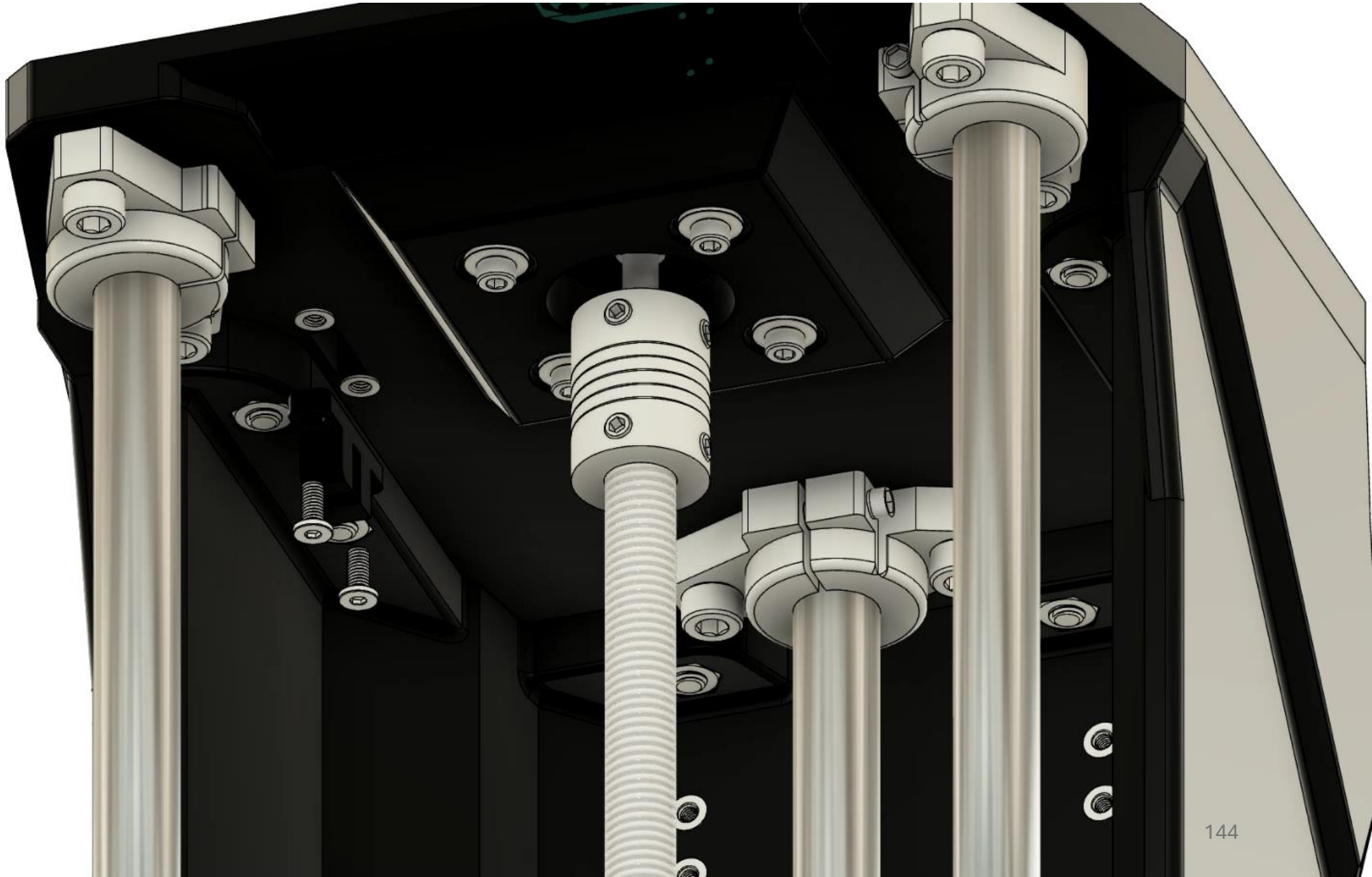
6x M5x15
6x M5 nut
4x M3x10

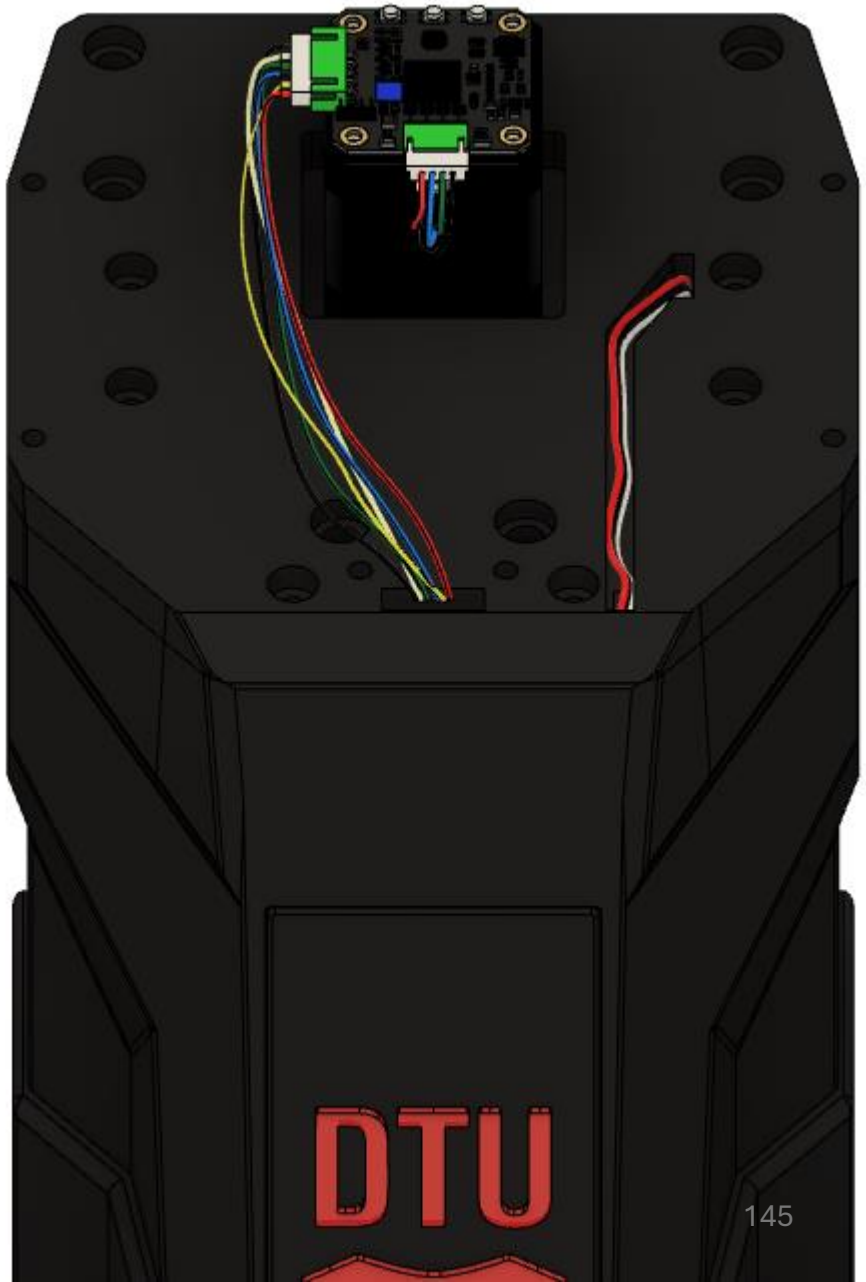


4x M3x15
4x M3 washer
6x M5x15
6x M5 nut



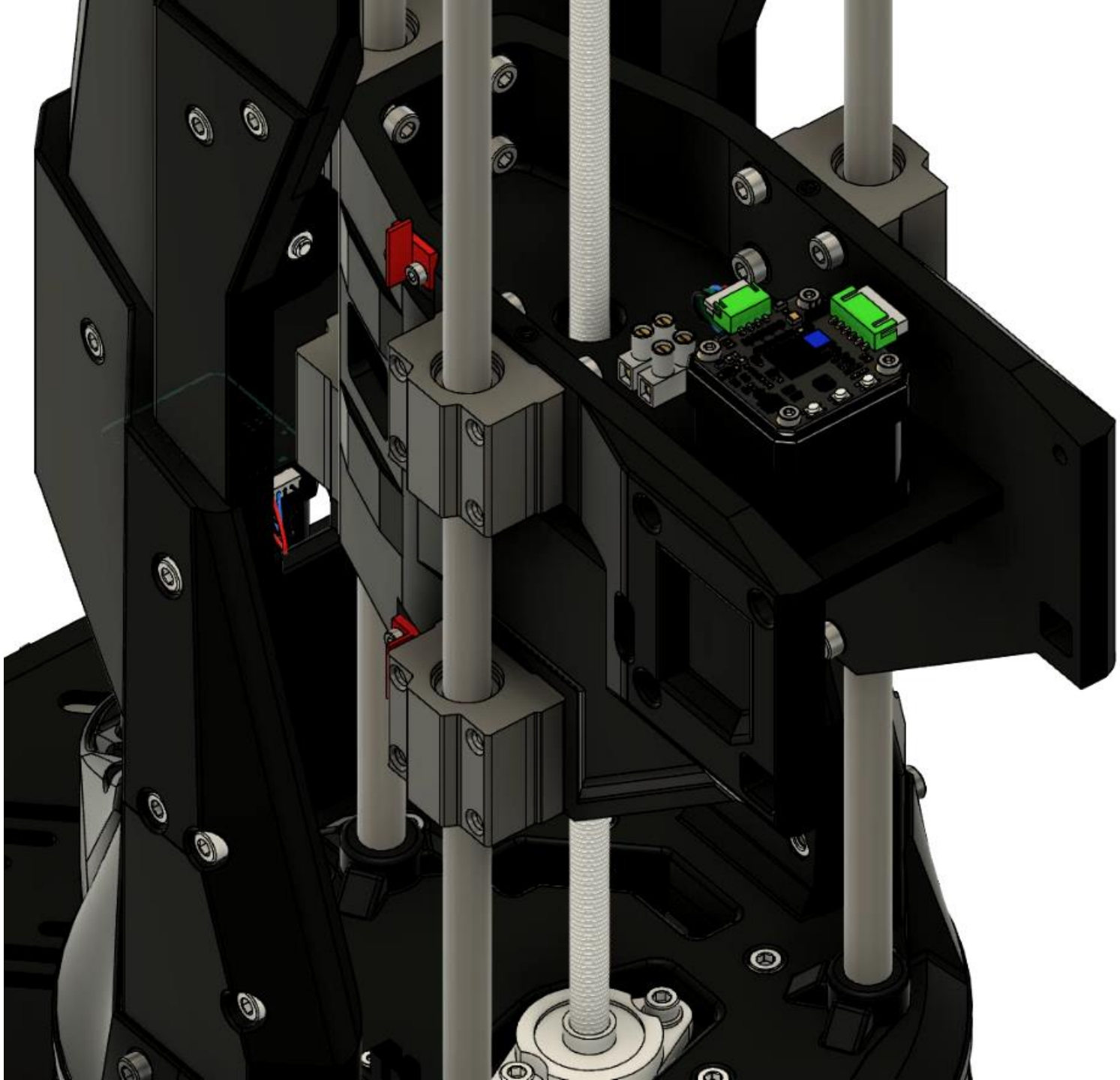
1x TCST2103
2x M3x8



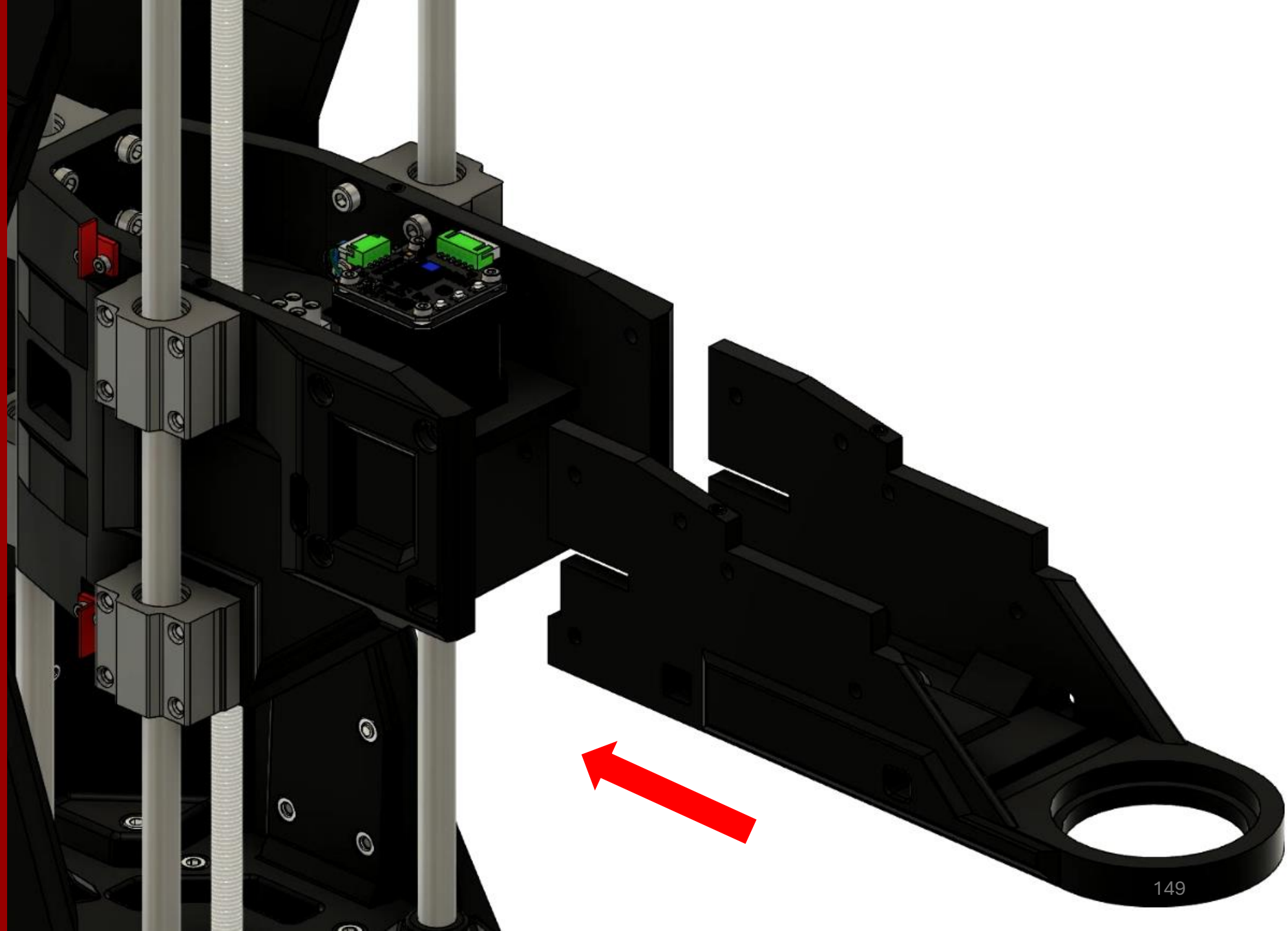

DTU

4x M4x10

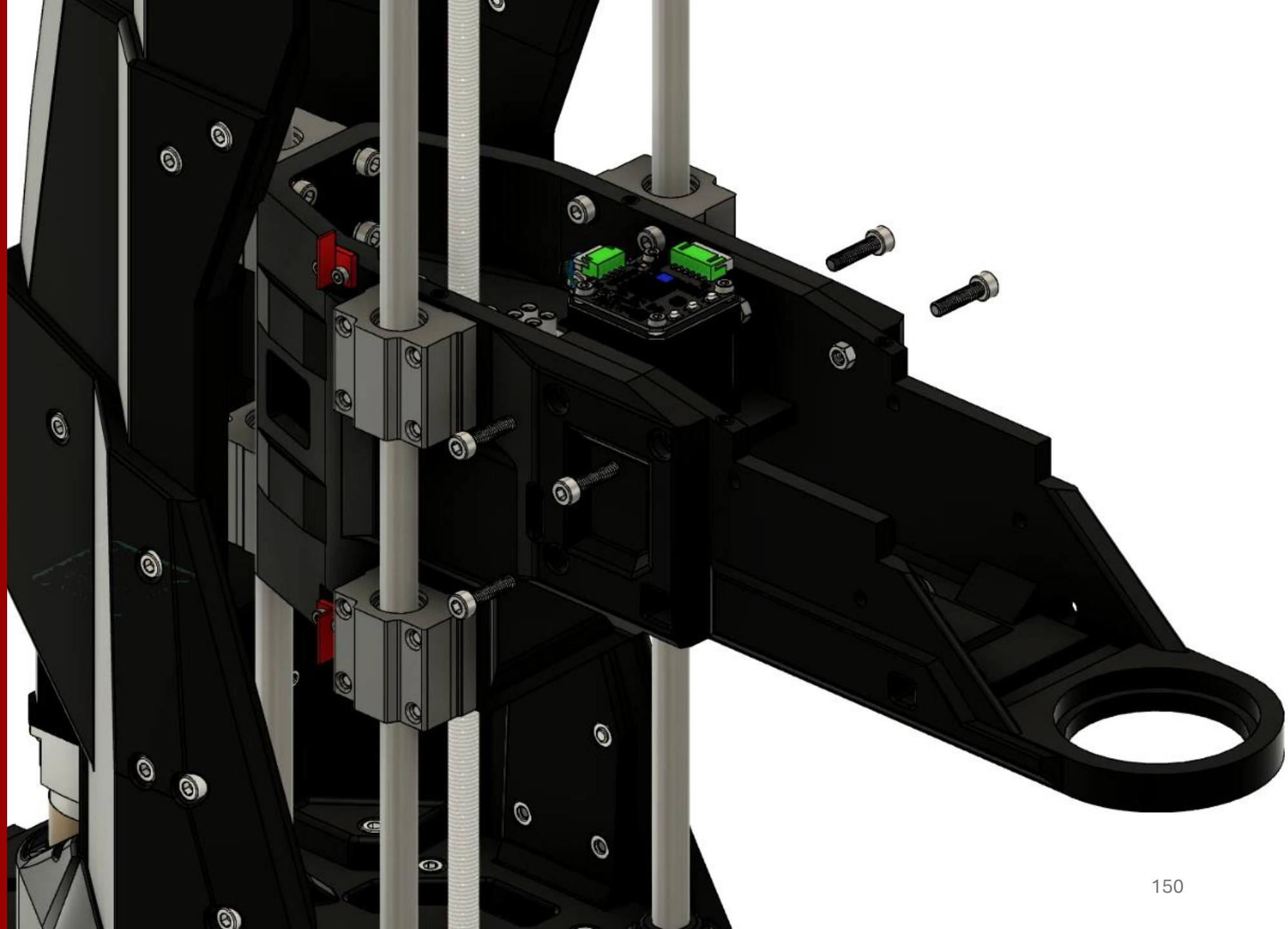
1x Nema 17 (17HS15-1504S1)
1x MKS Servo42C
4x M3x10
4x Distance washer



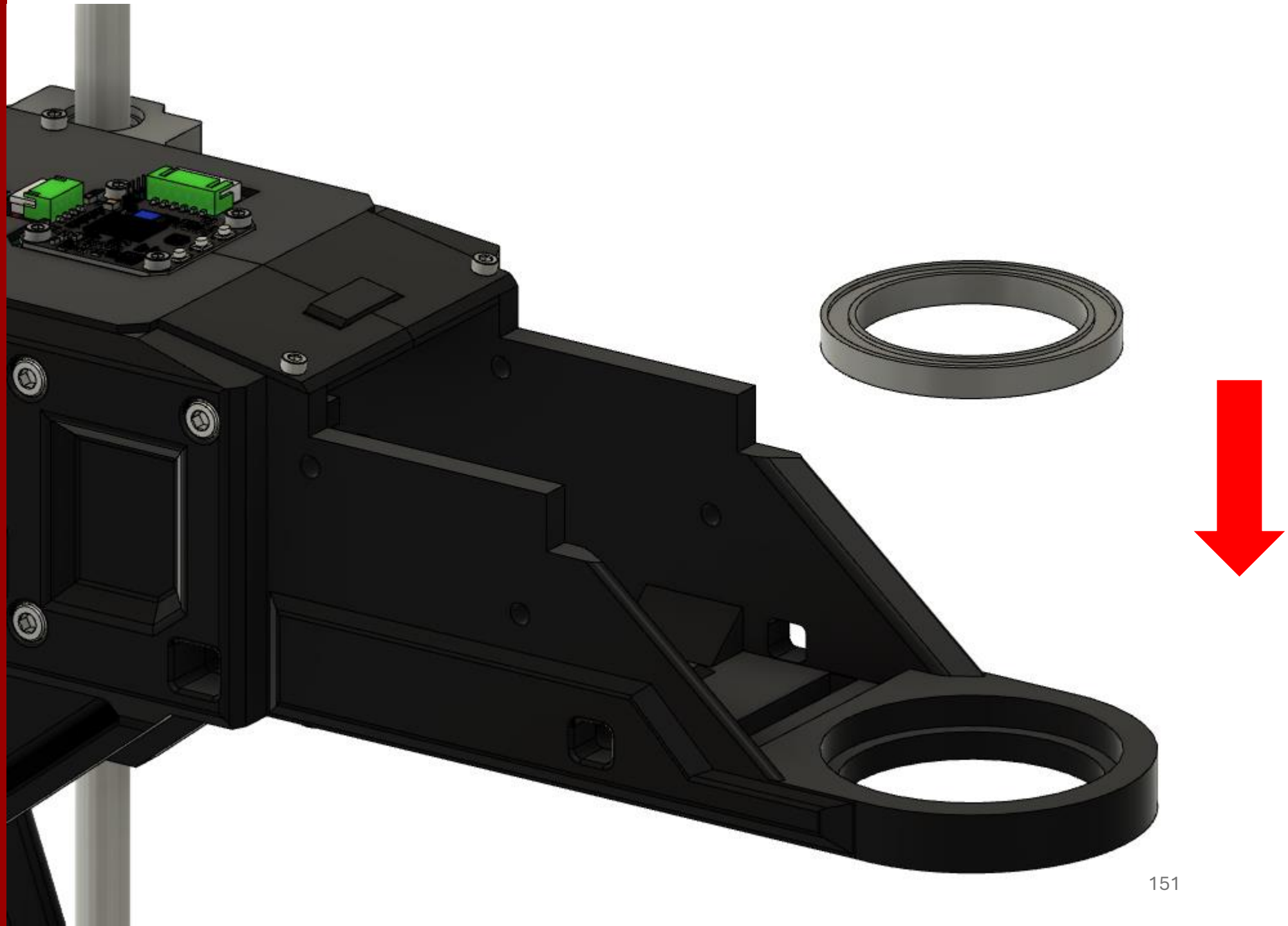
1x GT2 16T Pulley
4x M3x10



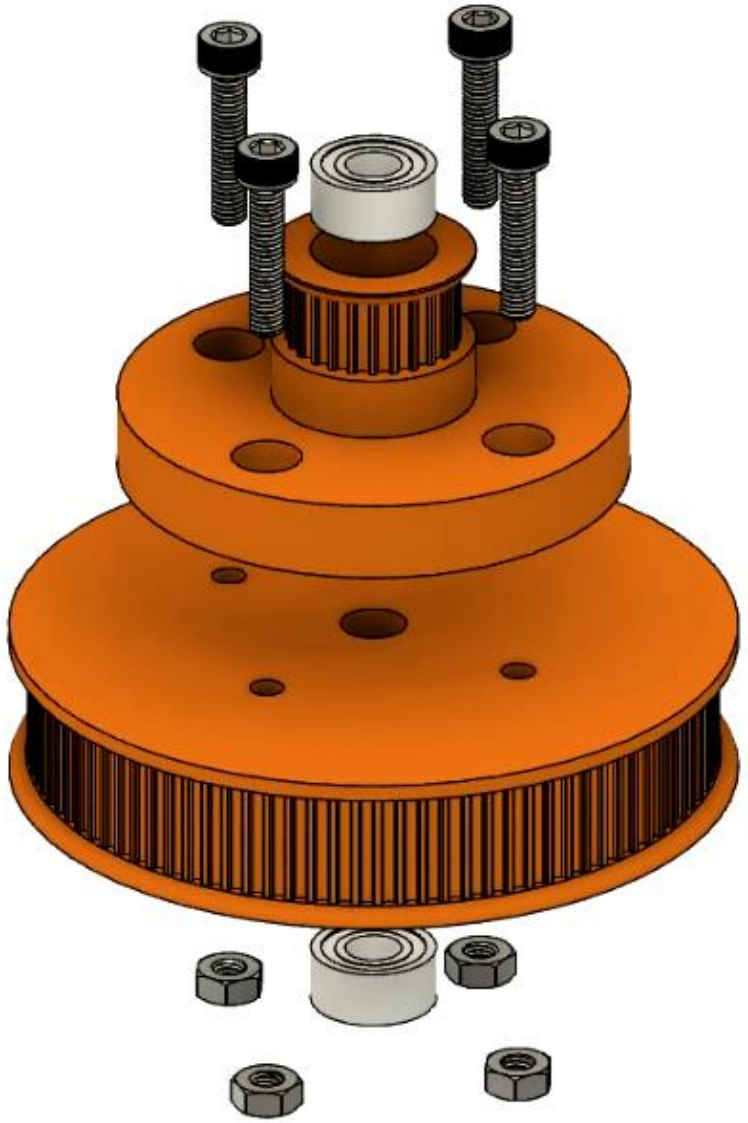
6x M5x20
6x M5 nut



1x Ball bearing 50x65x7

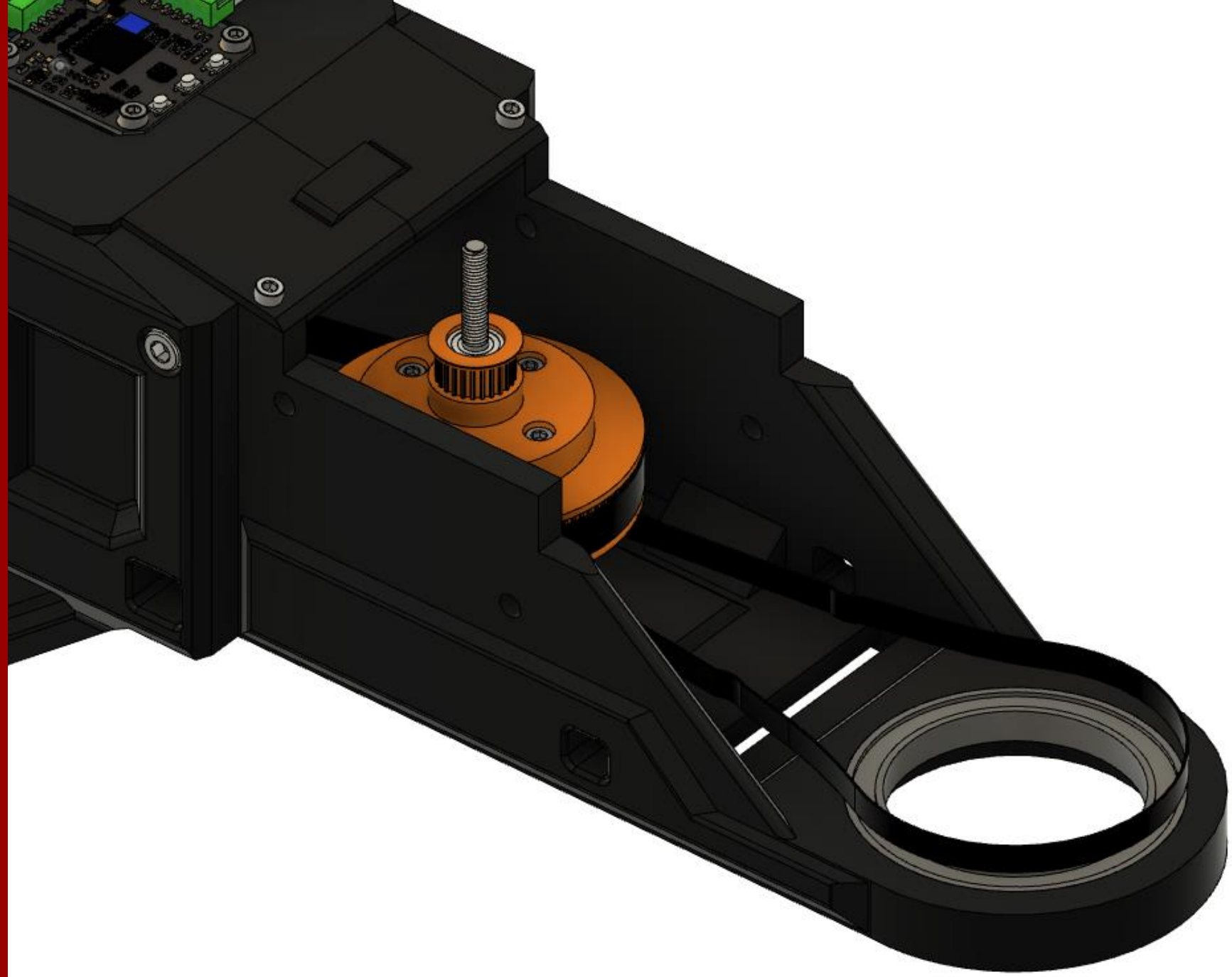


2x Ball bearing 5x11x5
4x M3x15
4x M3 nut

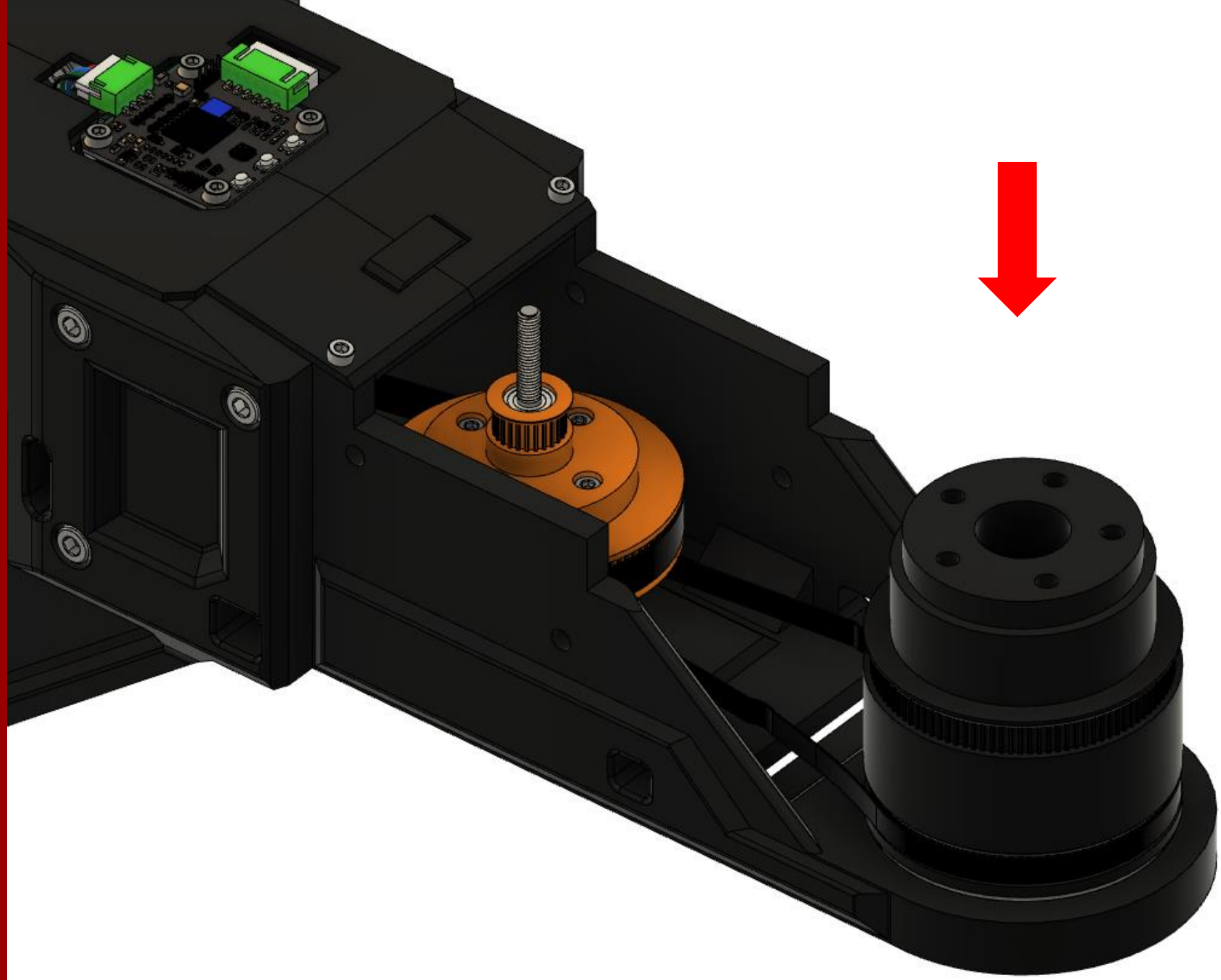


1x M5x75
1x GT2 belt 10x294
1x GT2 belt 6x400

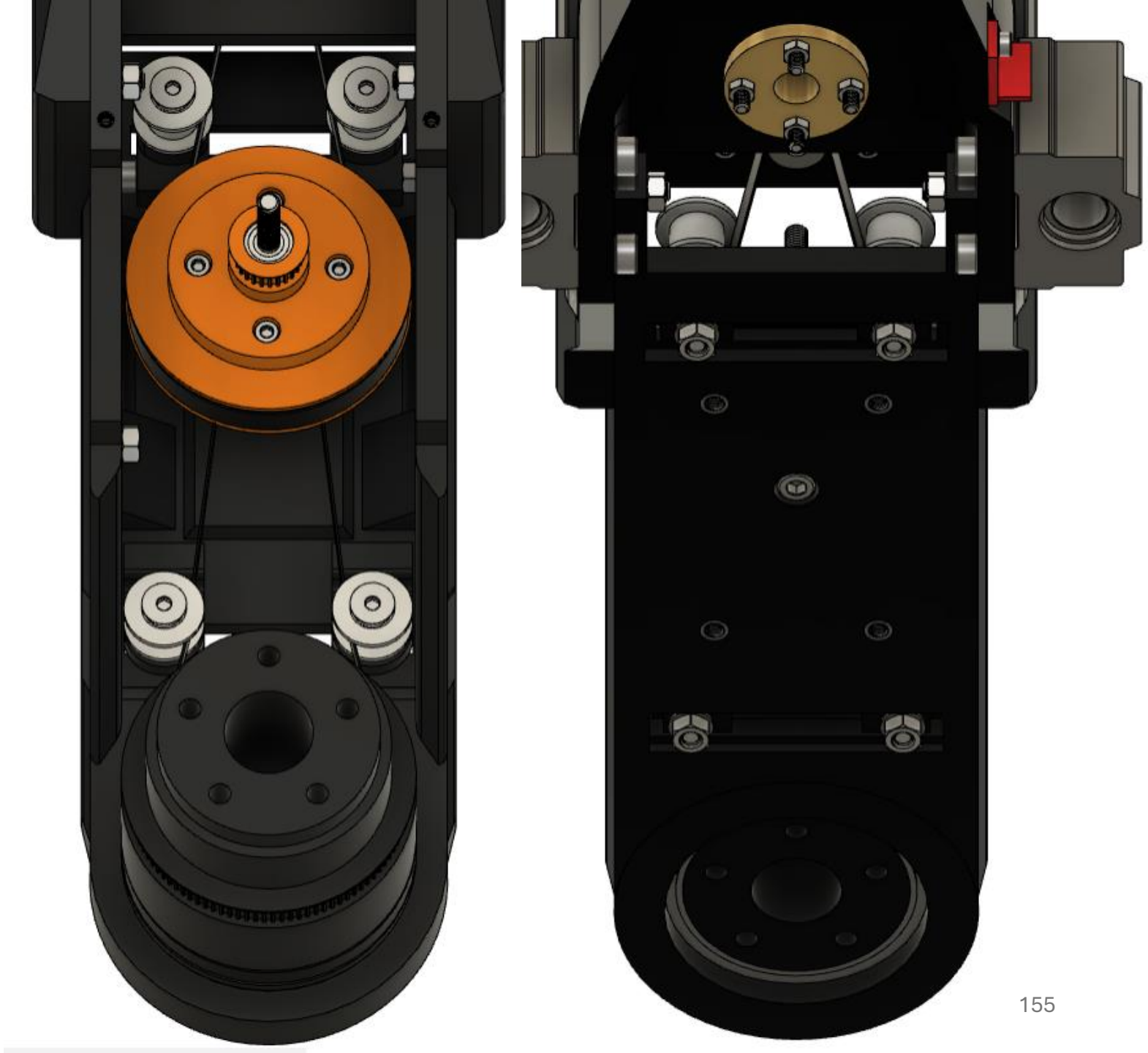
Add the belts onto the 3d printed pulley
Position the pulley inside the arm and insert the screw



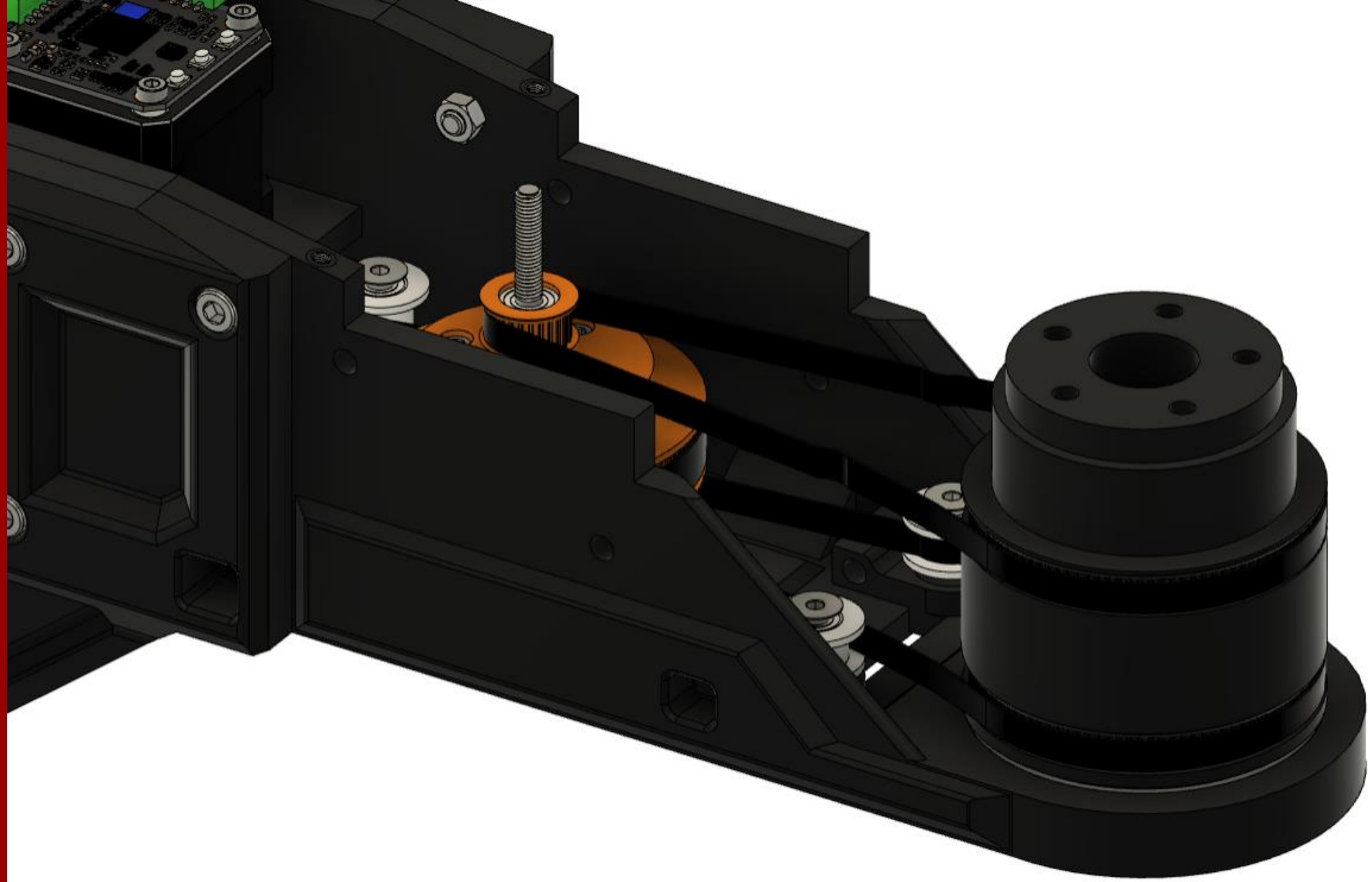
2x Ball bearing 5x11x5
4x M3x15
4x M3 nut



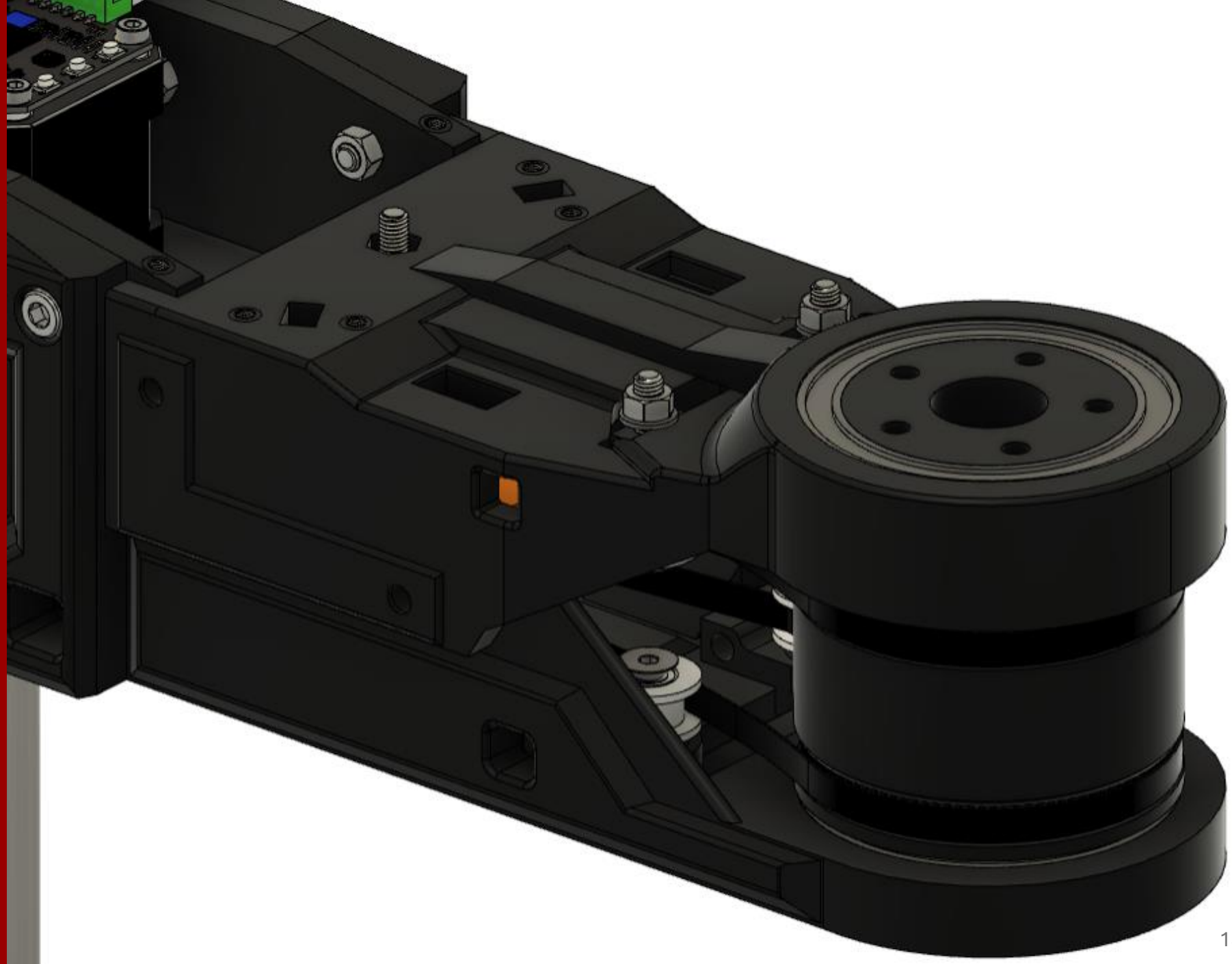
Belt tensioning system:
4x Pulley
2x M5x30 flat head screw
2x M5x40 flat head screw
4x M5 washer
4x M5 nut



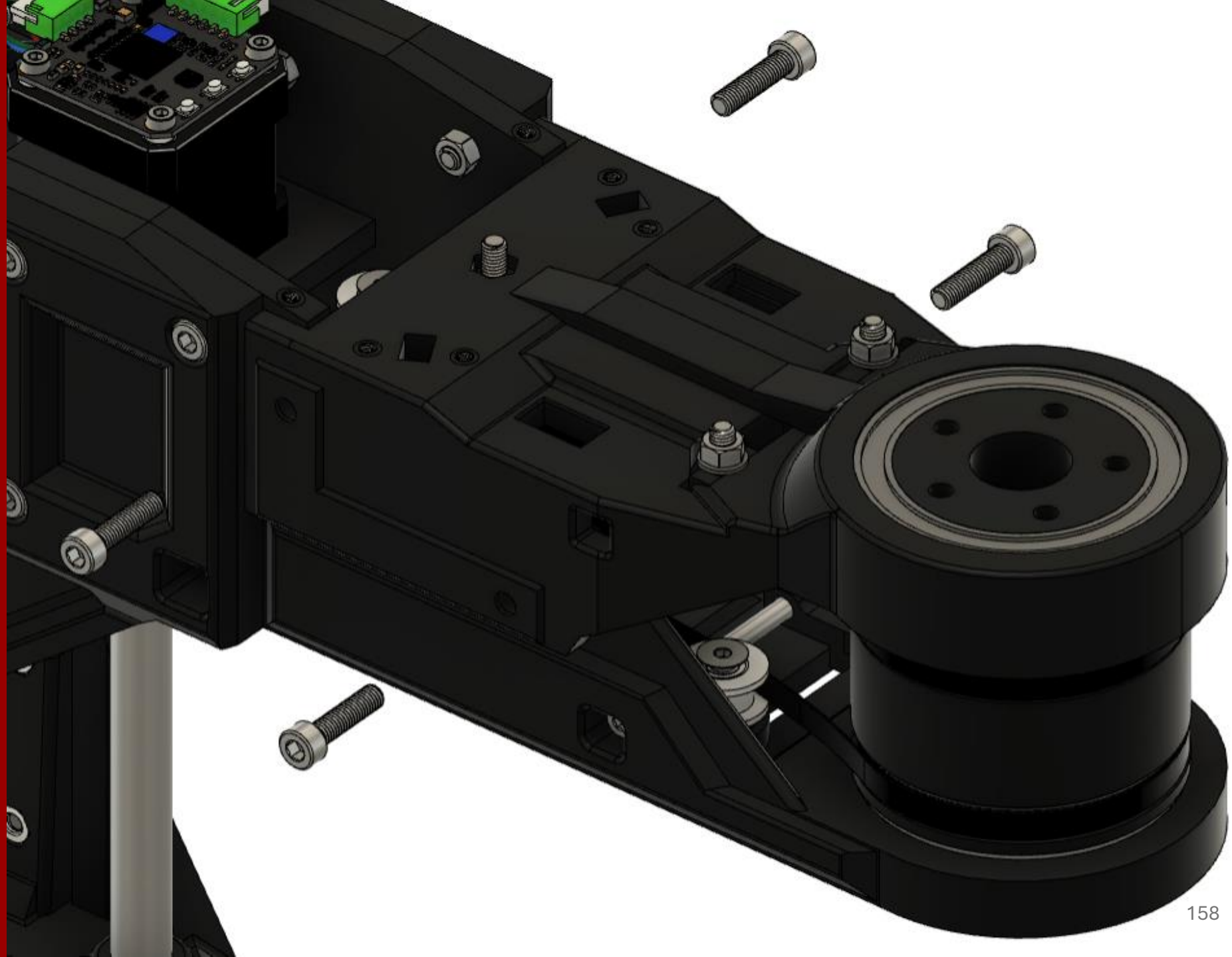
1x GT2 belt 6x400



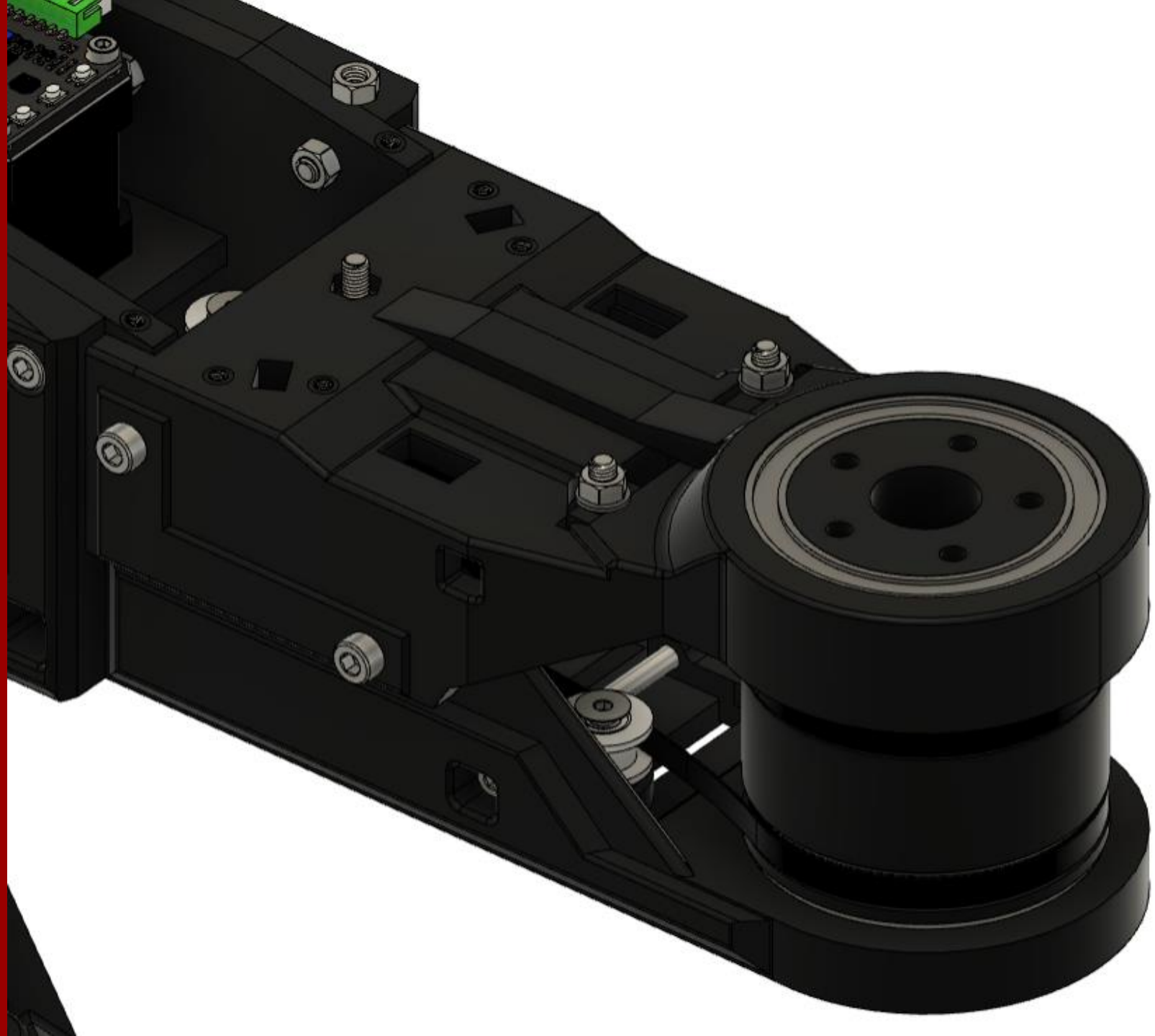
1x Ball bearing 50x65x7
2x Pulley
2x M5x35 flat head screw
2x M5 washer
2x M5 nut



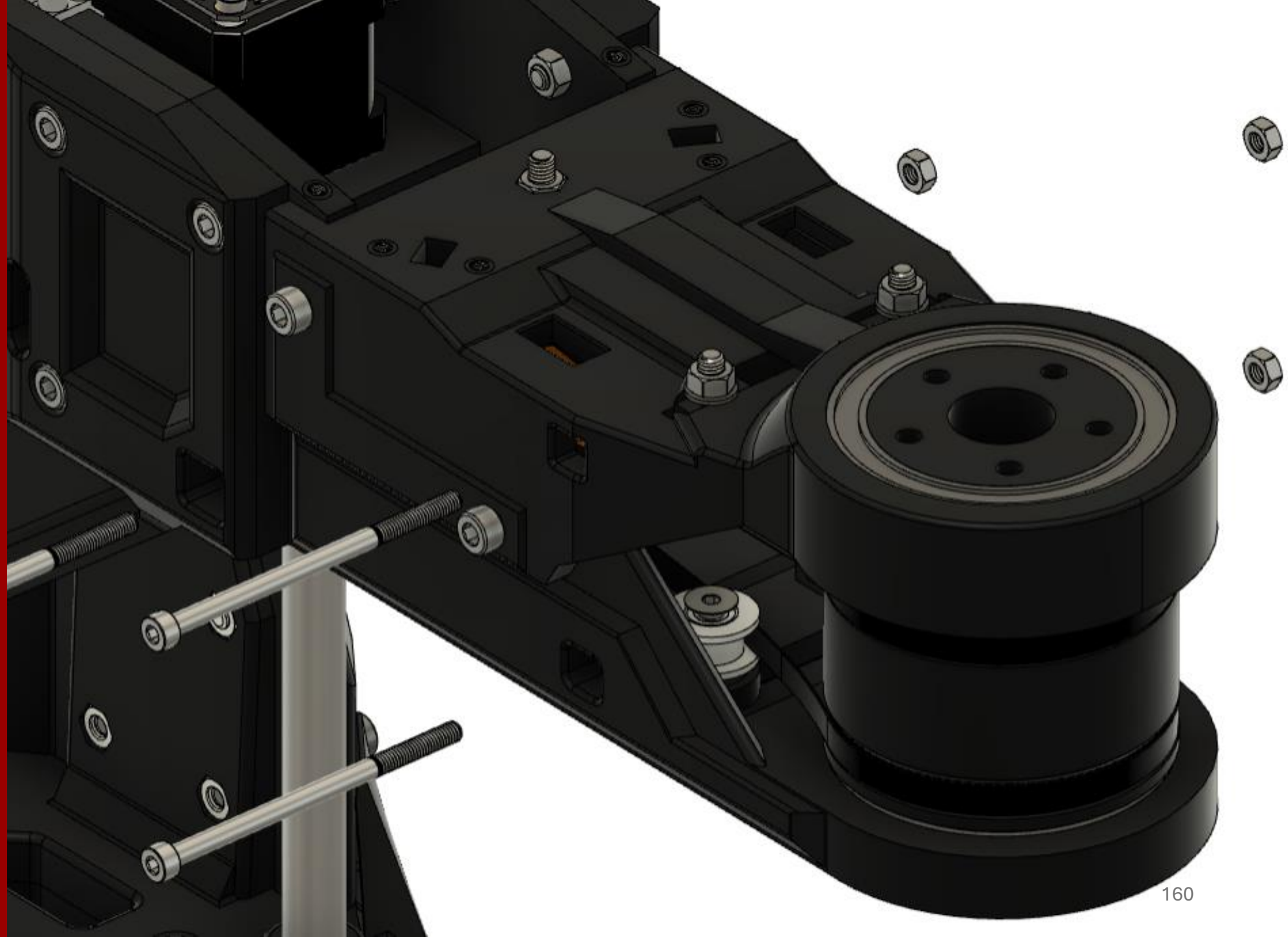
4x M5x20
4x M5 nut



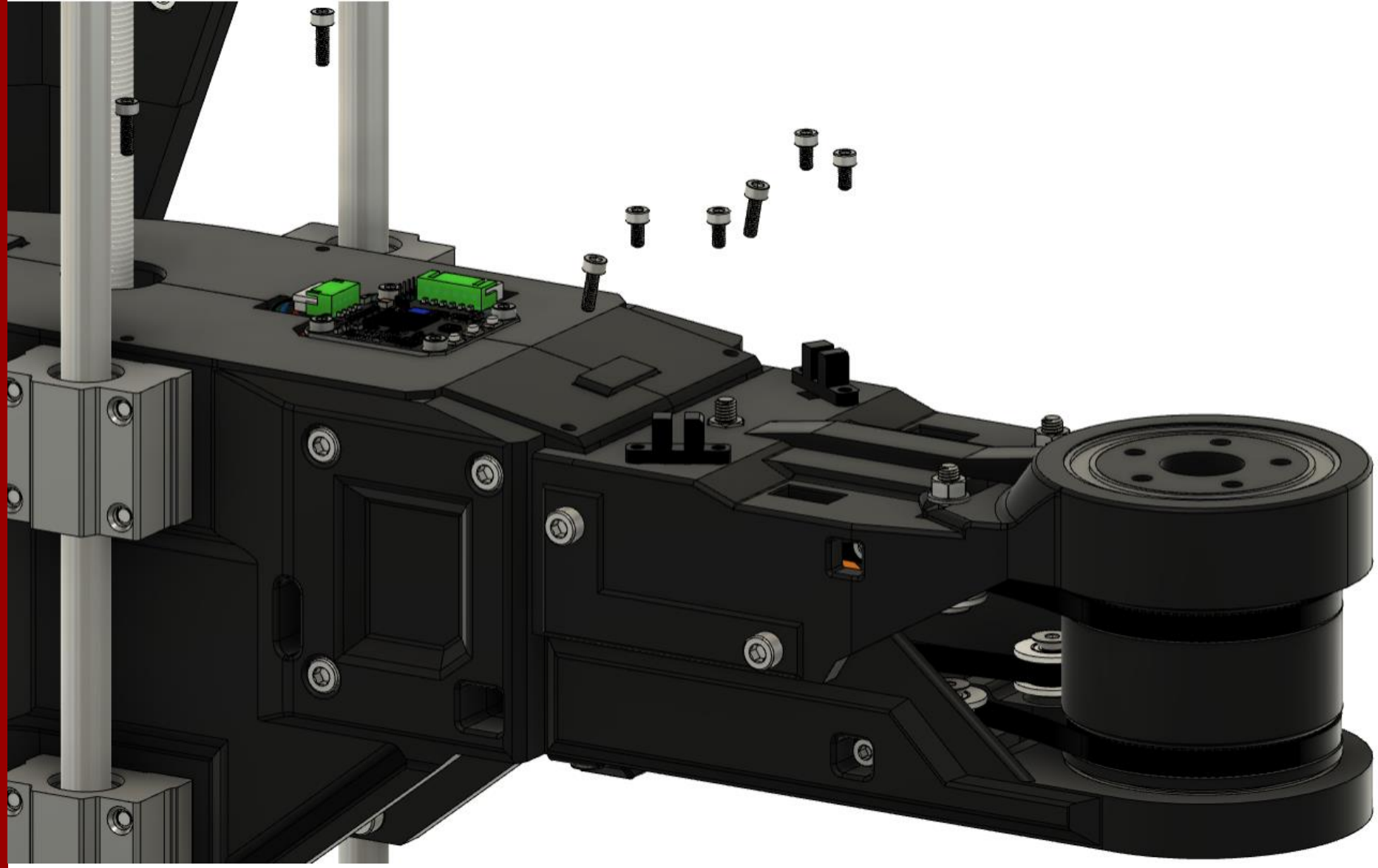
1x M5 nut



3x M4x70
3x M4 nut

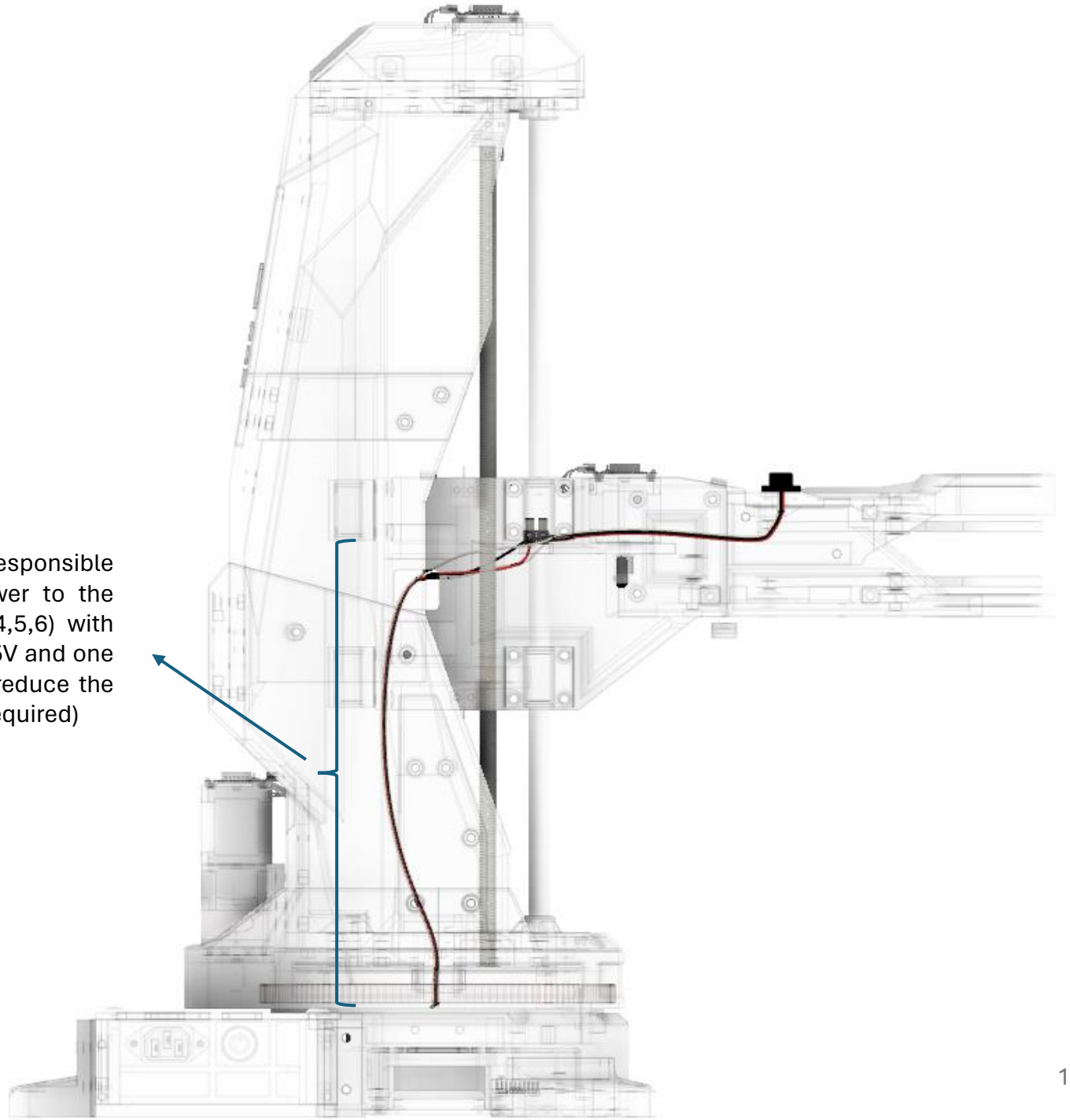


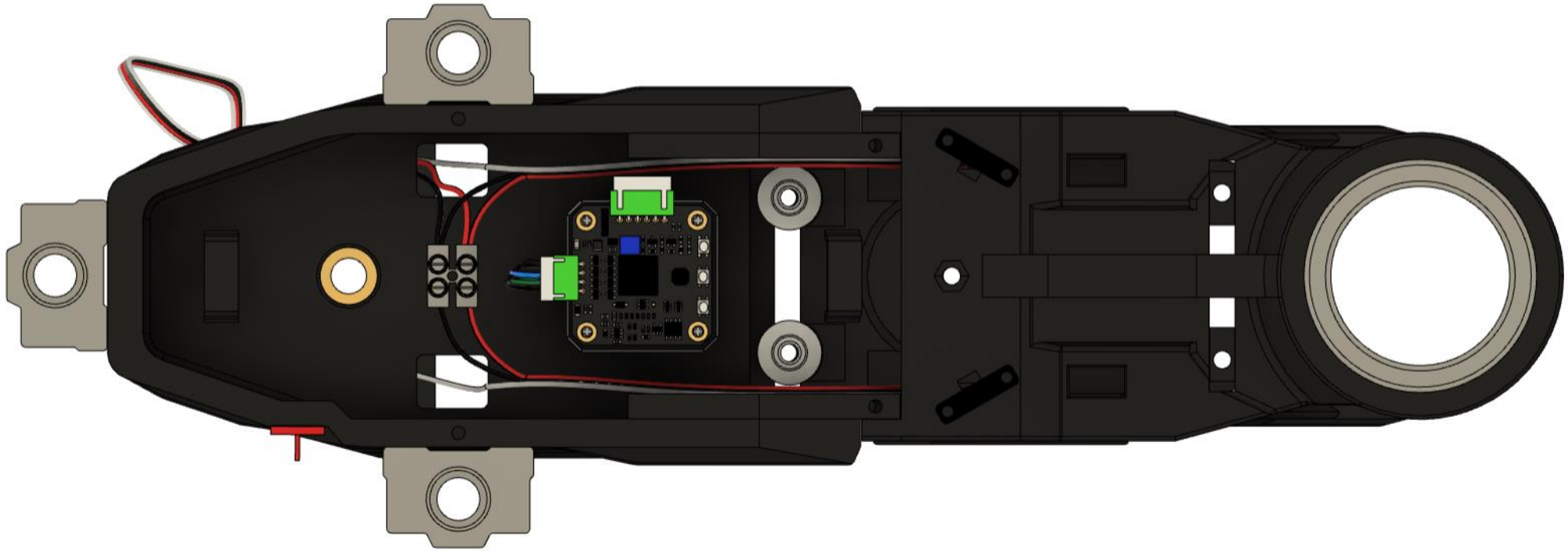
2x TCST2103
4x M3x10
4x M3x8

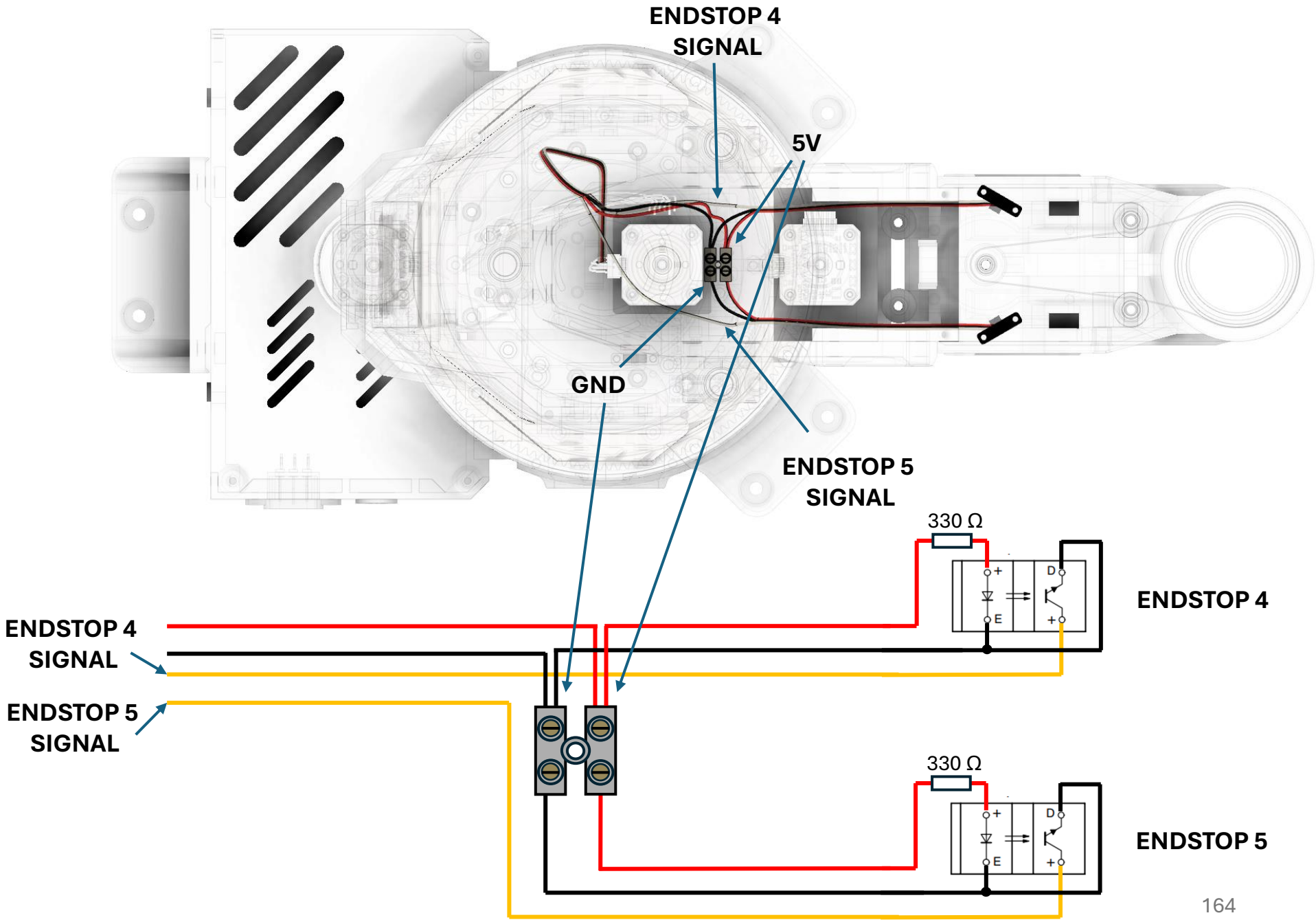


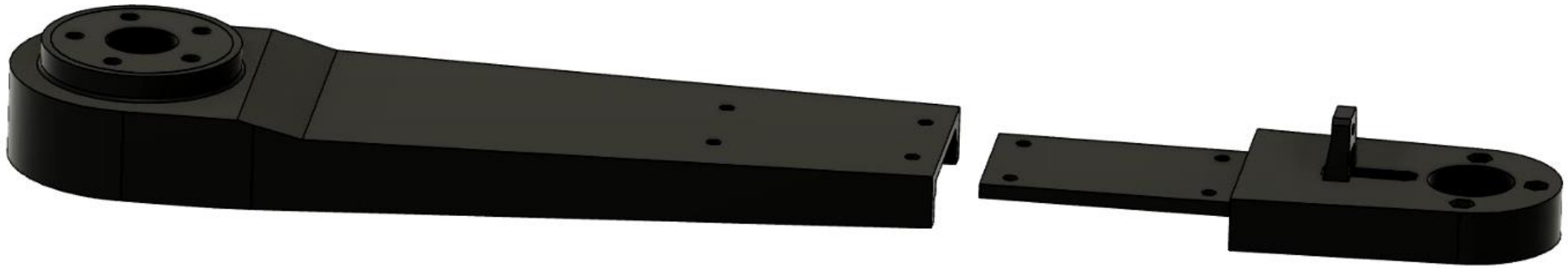
Note:
Do not mount the cover, this should be done later, after the wiring is completed

This segment is responsible for supplying power to the three endstops (4,5,6) with only one wire for 5V and one wire for GND (to reduce the amount of wires required)

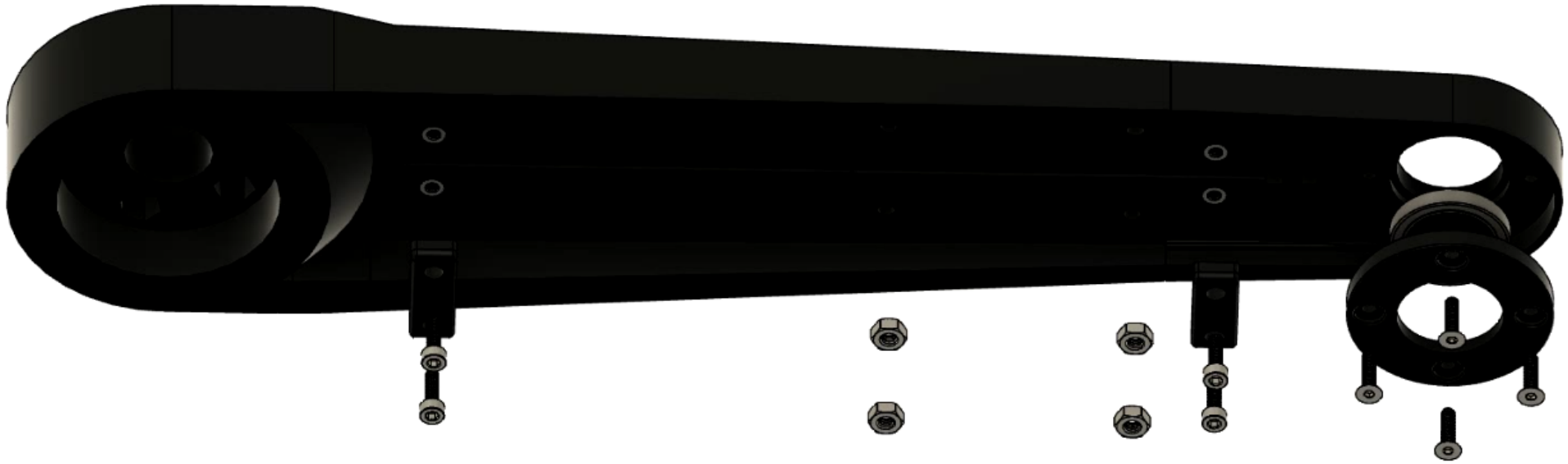




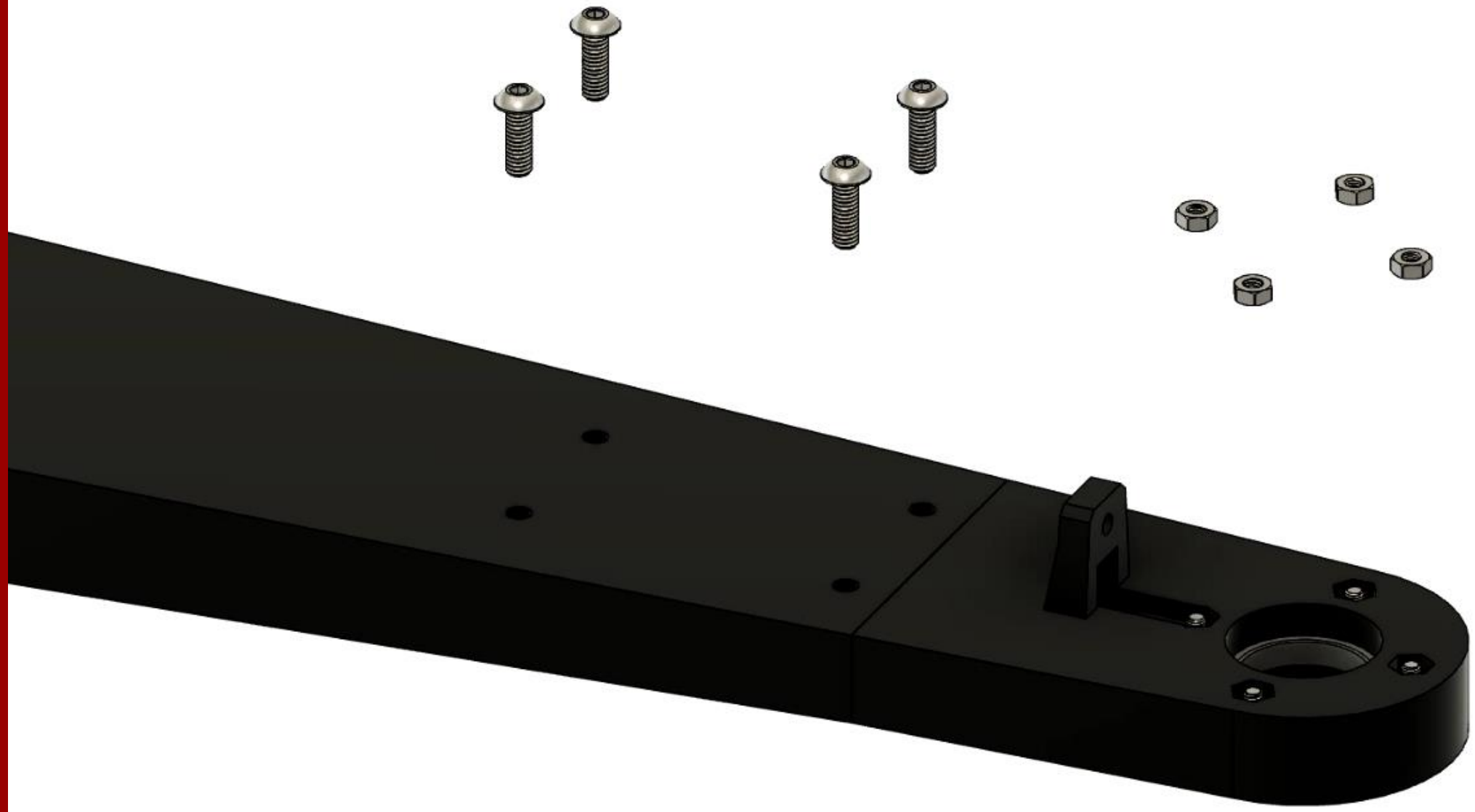

ENDSTOP 4
SIGNAL
5V
GND
ENDSTOP 5
SIGNAL
330 Ω
ENDSTOP 4
ENDSTOP 4
SIGNAL
ENDSTOP 5
SIGNAL
330 Ω
ENDSTOP 5



1x Ball bearing 20x27x4
4x M3x10
4x M3x10 flat head screw
4x M4 nut

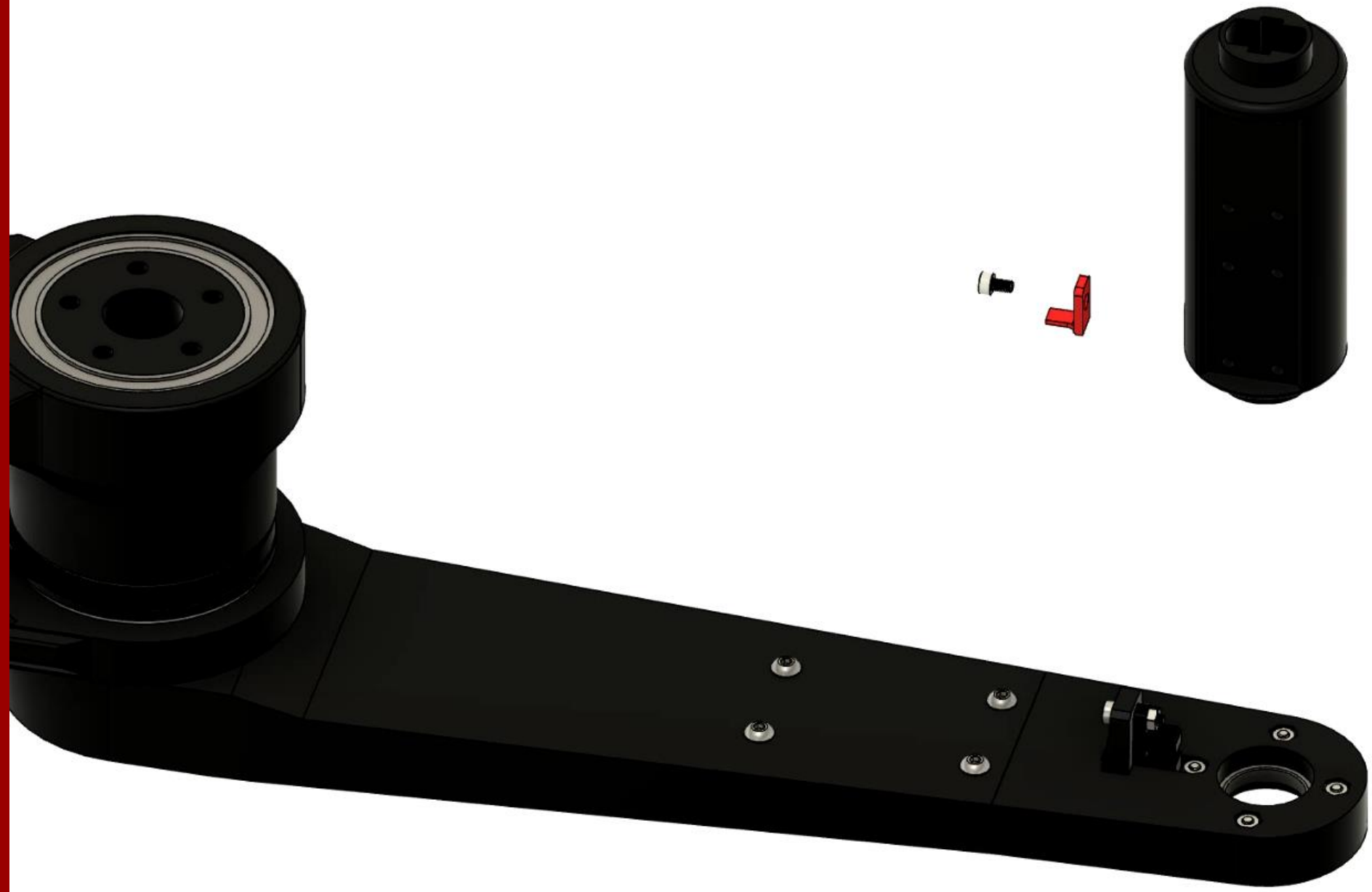


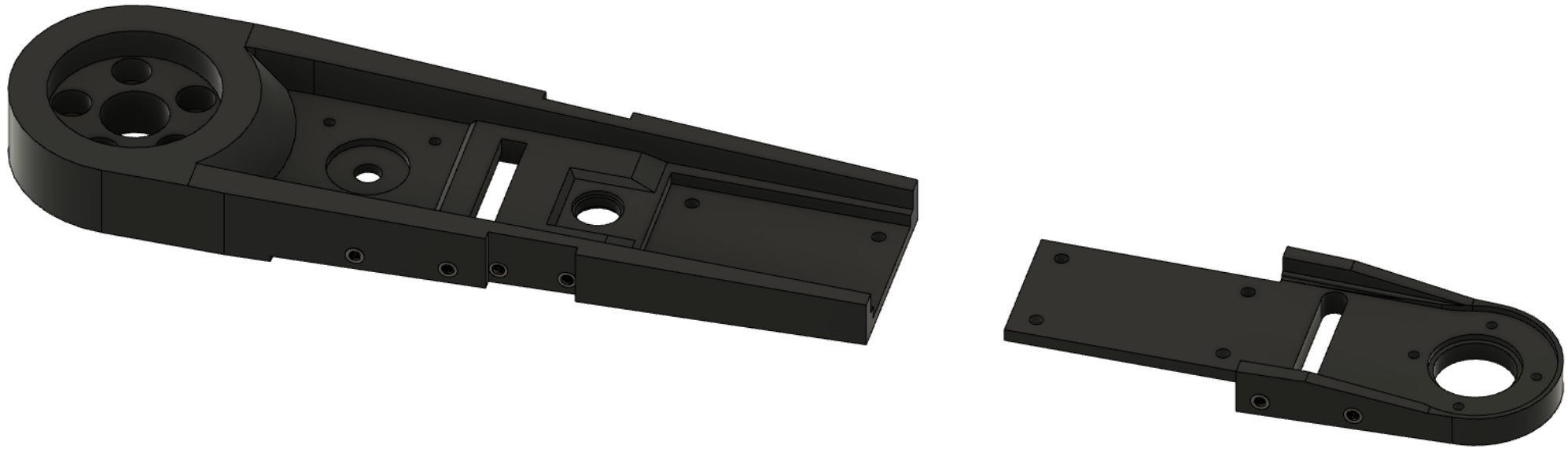
4x M4x15
4x M3 nut



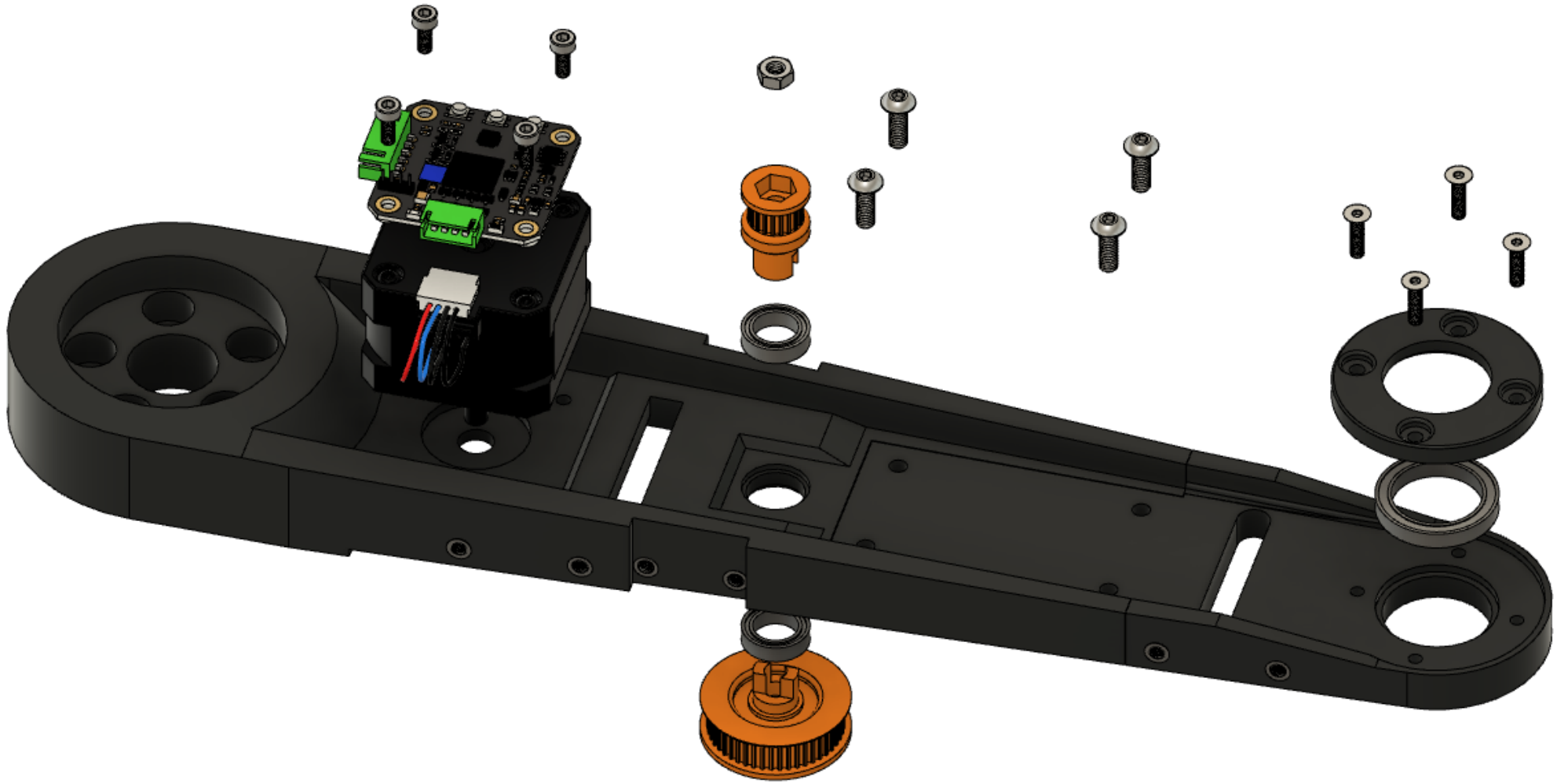
1x TCST2103
1x M3x10
1x M3 nut

1x M3x8

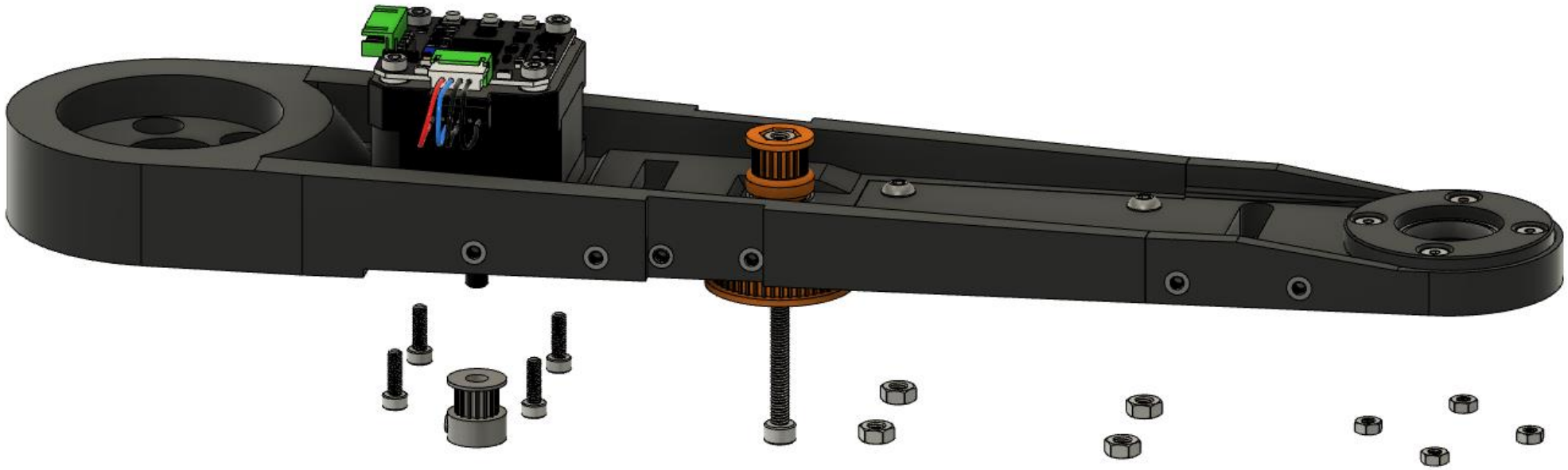




1x Nema 17 (17HE12-1204S)
1x MKS Servo42C
4x Distance washer
1x Ball bearing 20x27x4
2x Ball bearing 10x15x4
4x M3x10 flat head screw
4x M4x15
4x M3x10
1x M4 nut

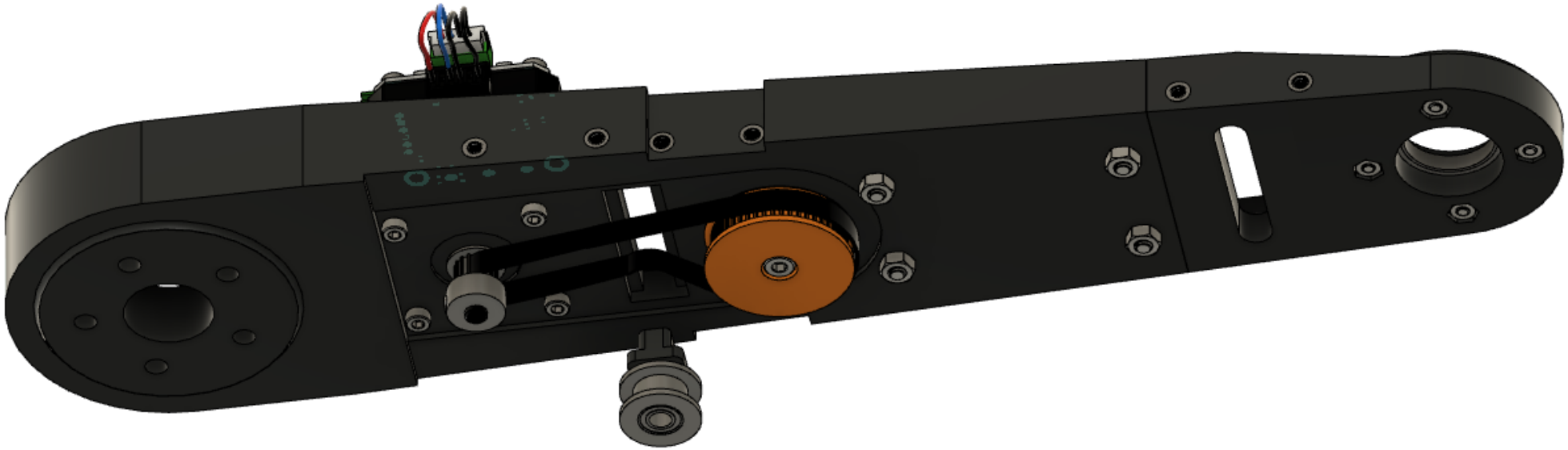


1x GT2 16T Pulley
4x M3 nut
4x M4 nut
1x M4x30
4x M3x10

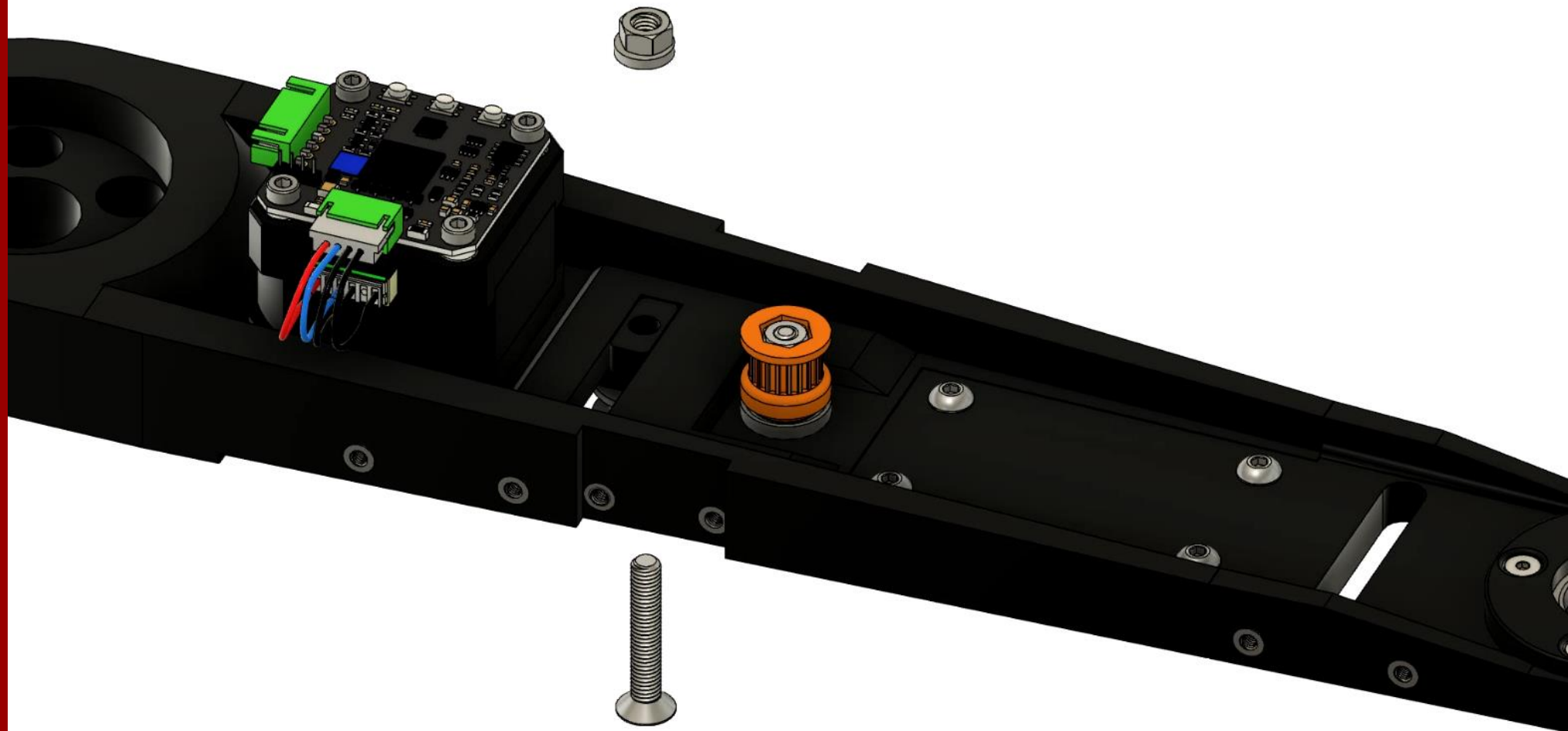


1x GT2 Belt 6x202

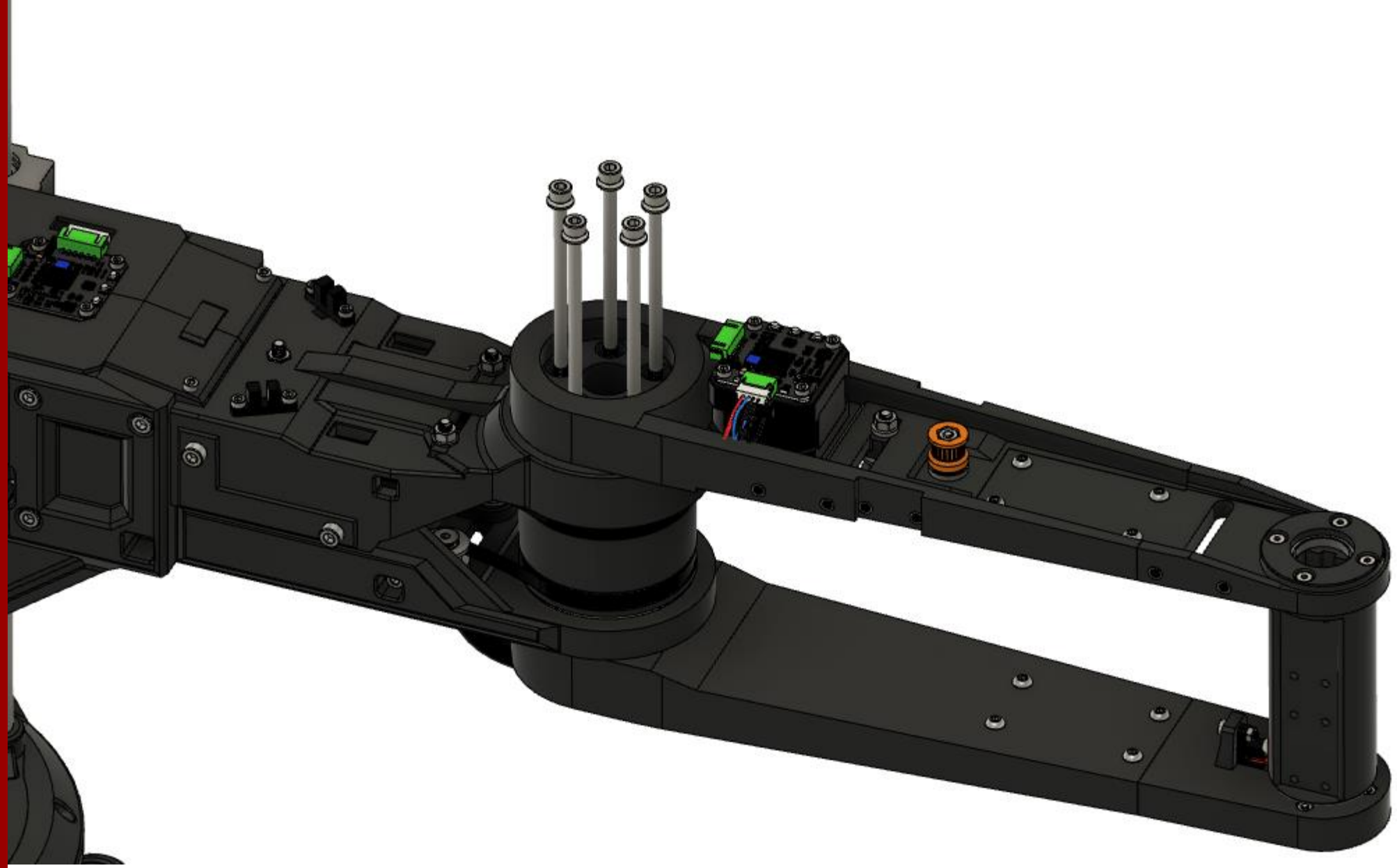
1x Pulley

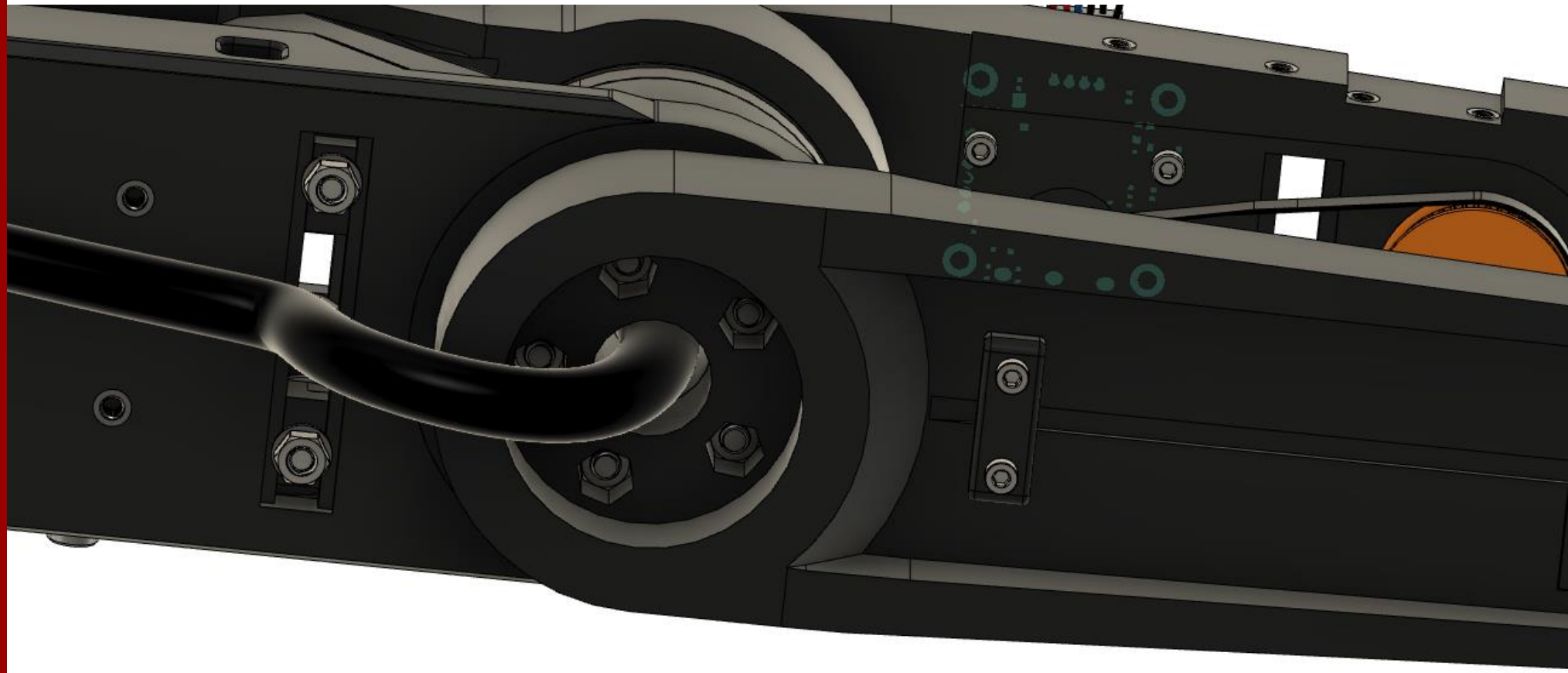


1x M5x30 flat head screw
1x M5 nut
1x M5 washer



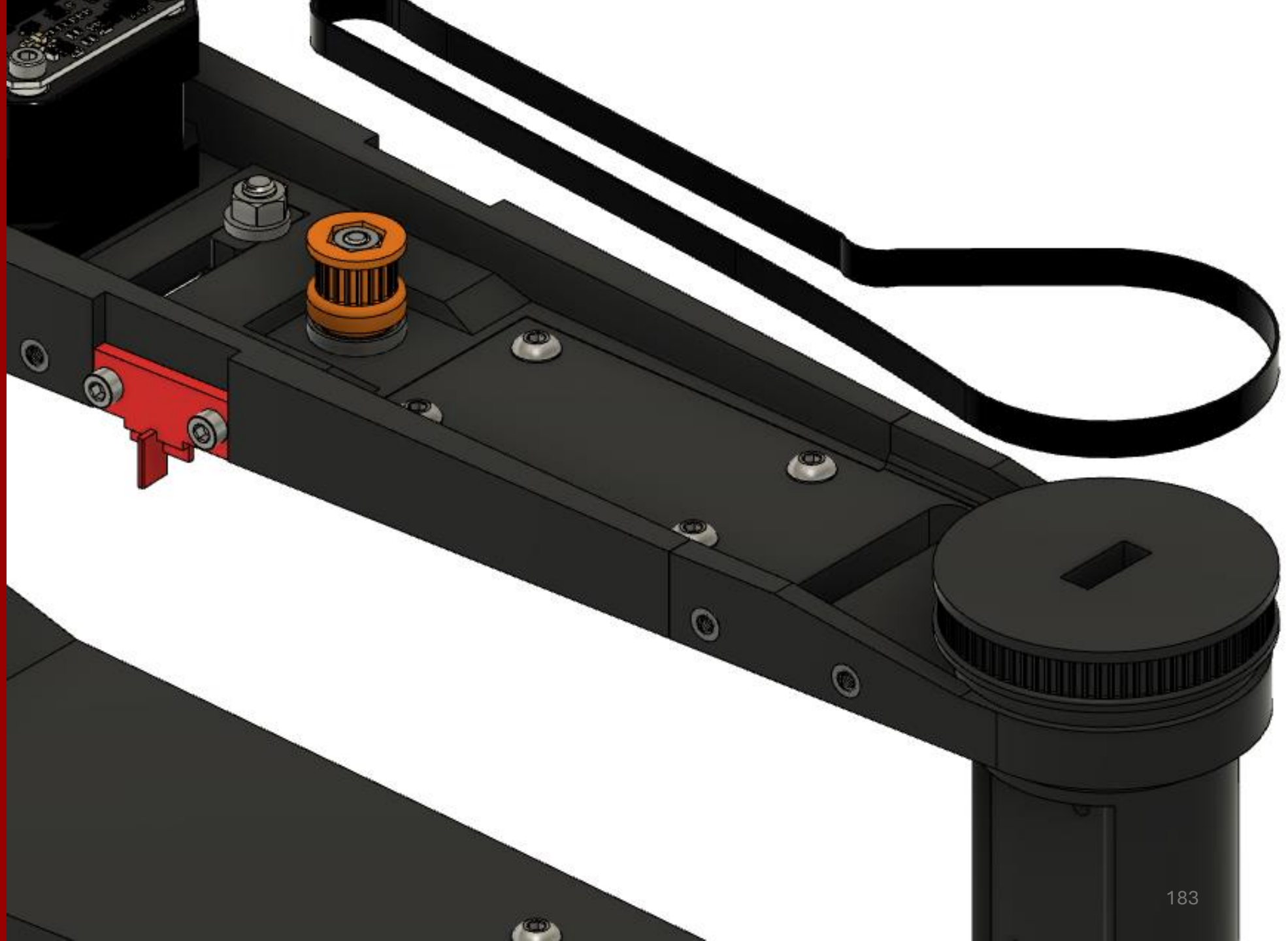
5x M5x100
5x M5 washer



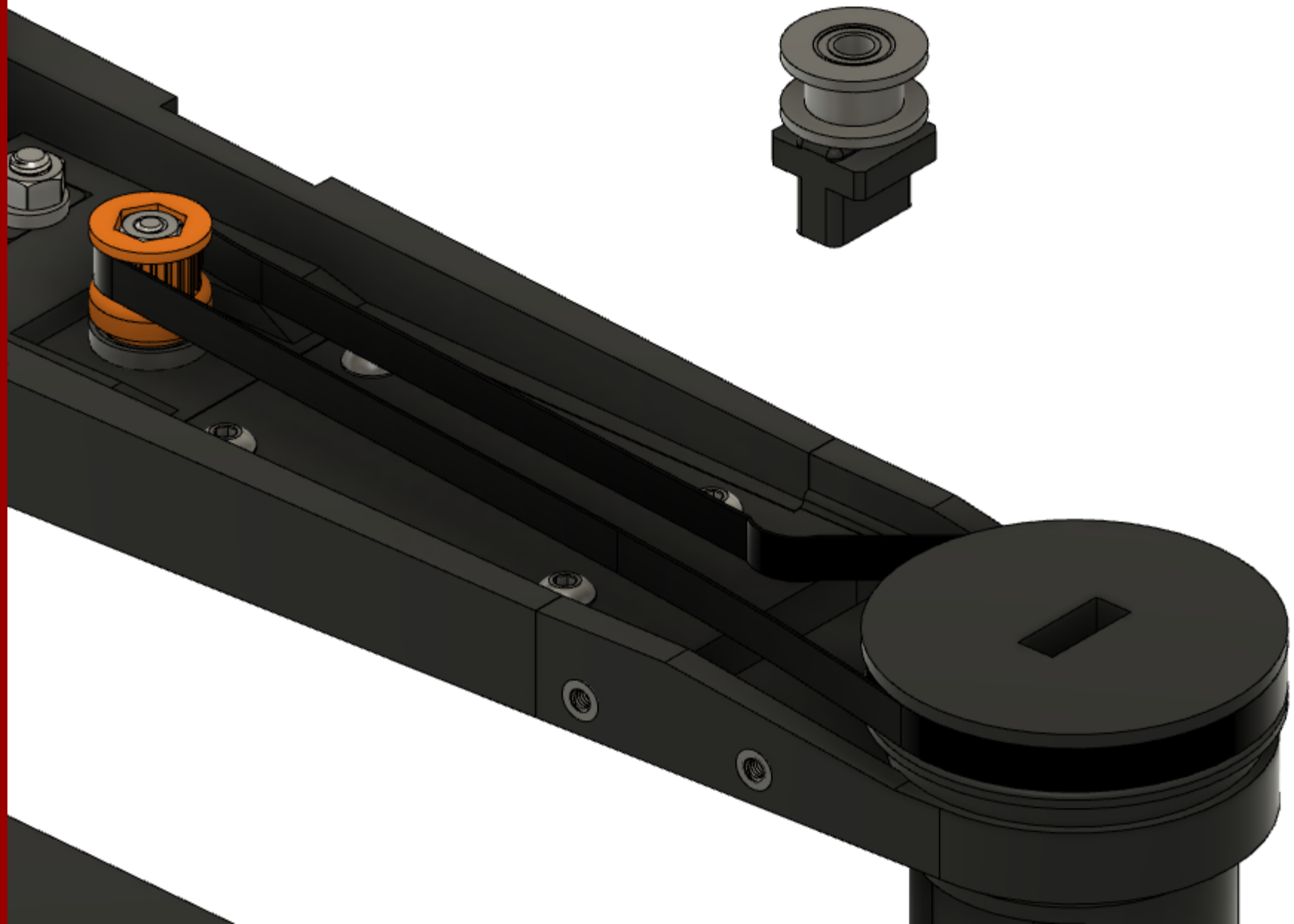


4x M3x8

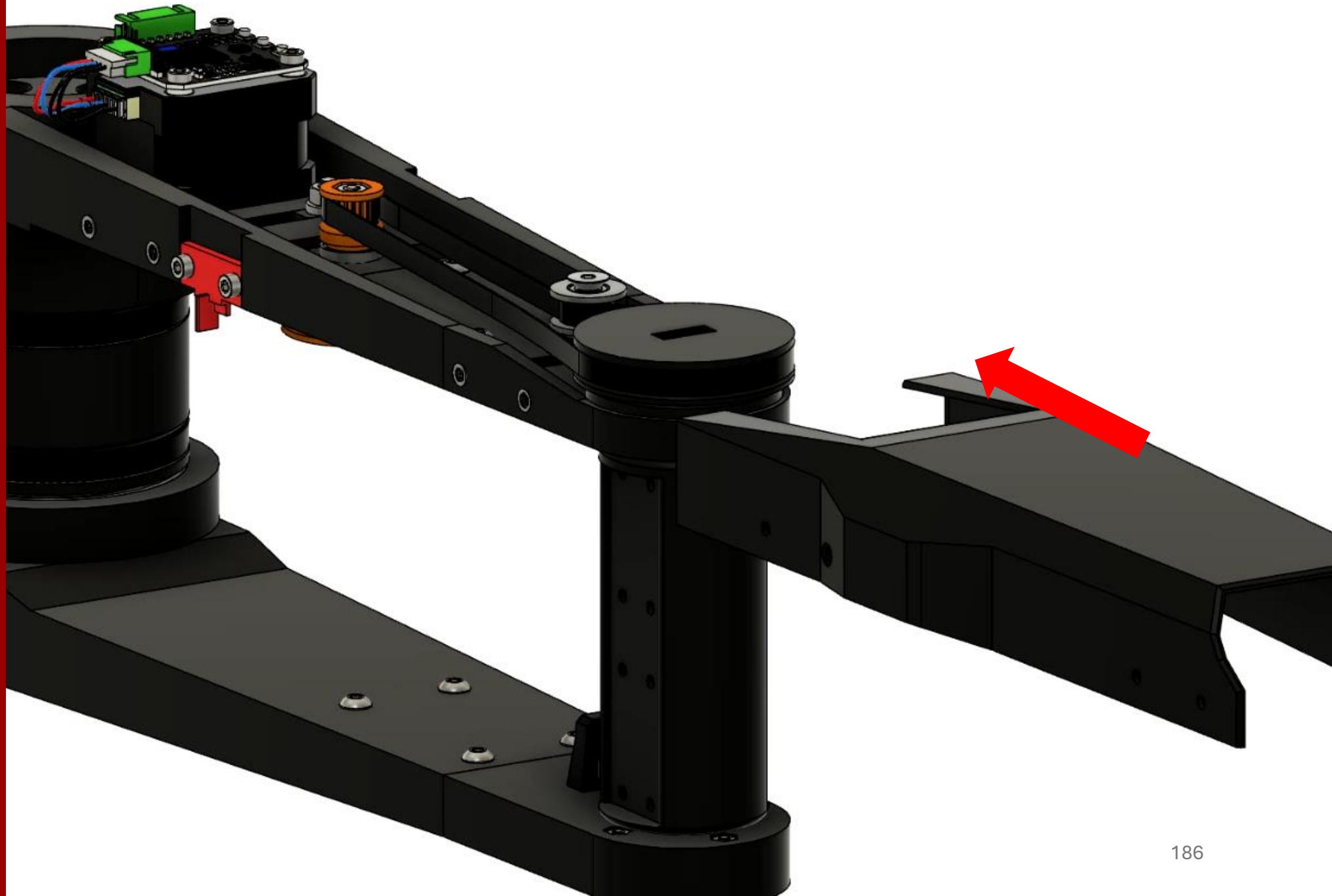
1x GT2 Belt 6x406



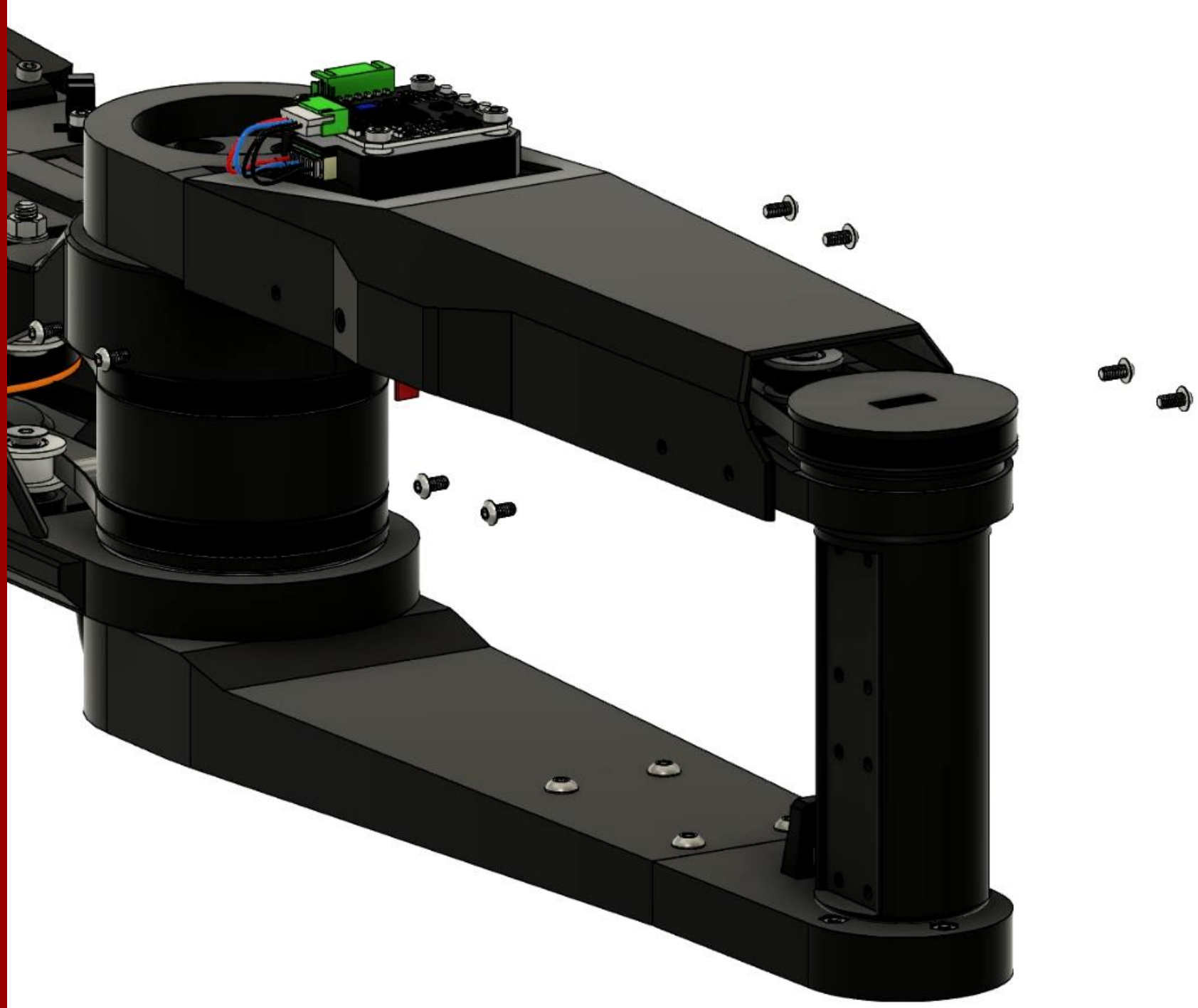
1x Pulley



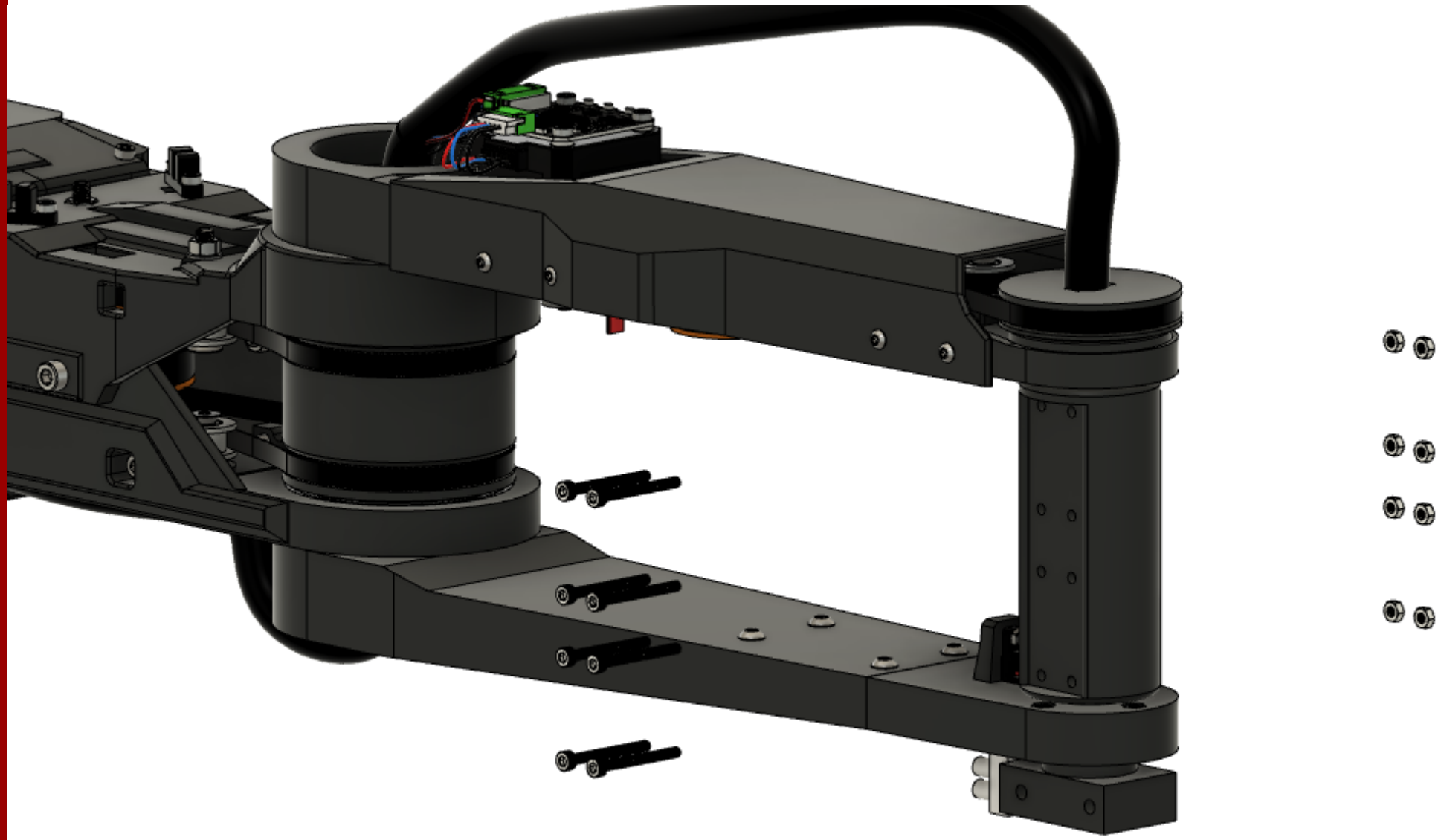
1x M5x35 flat head screw
1x M5 washer
1x M5 nut

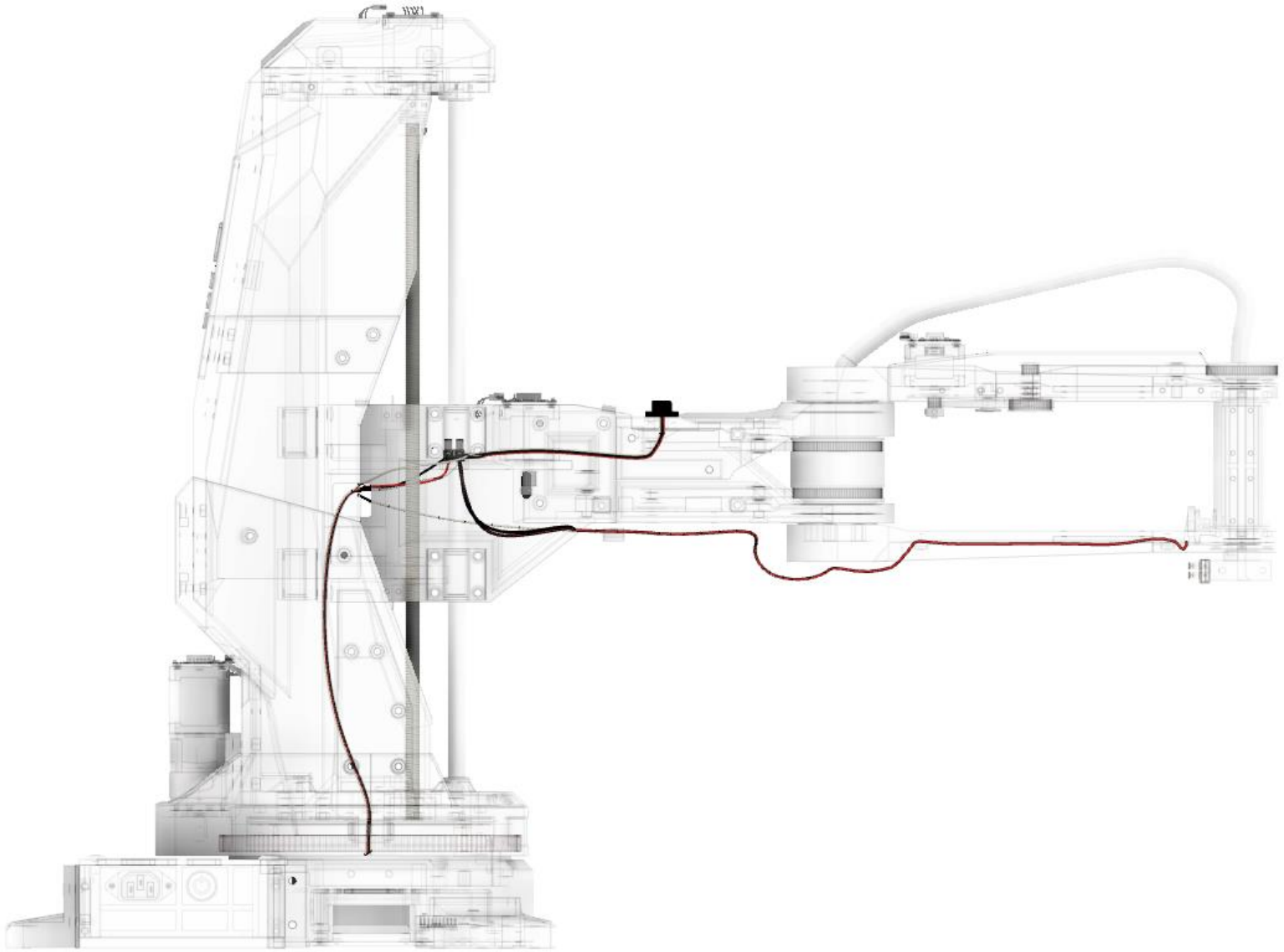


4x M3x8

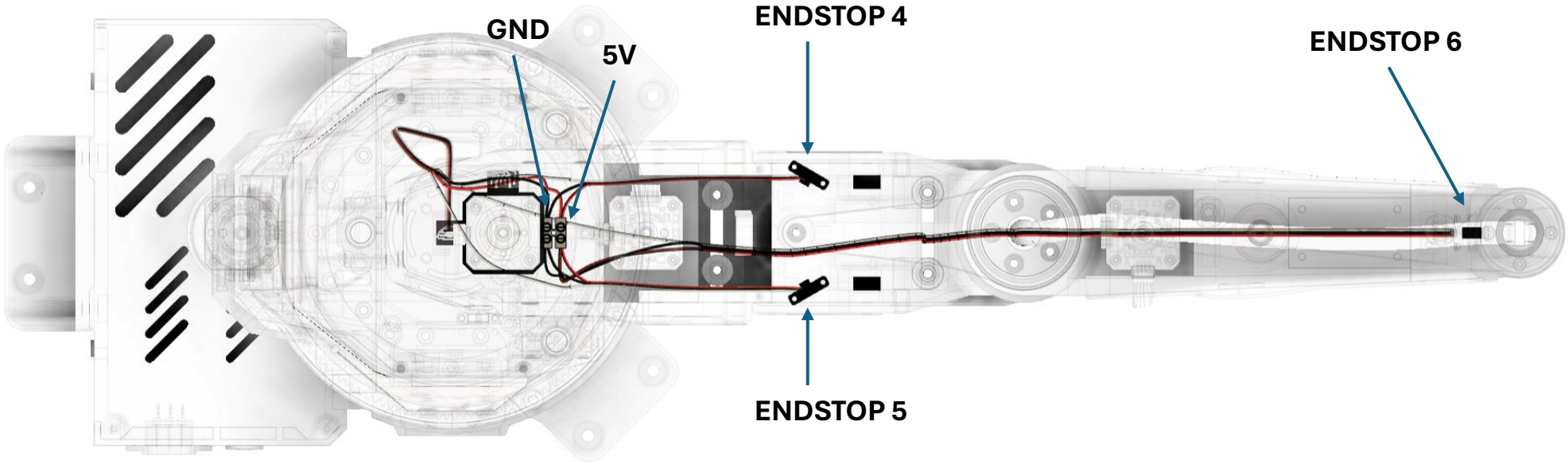


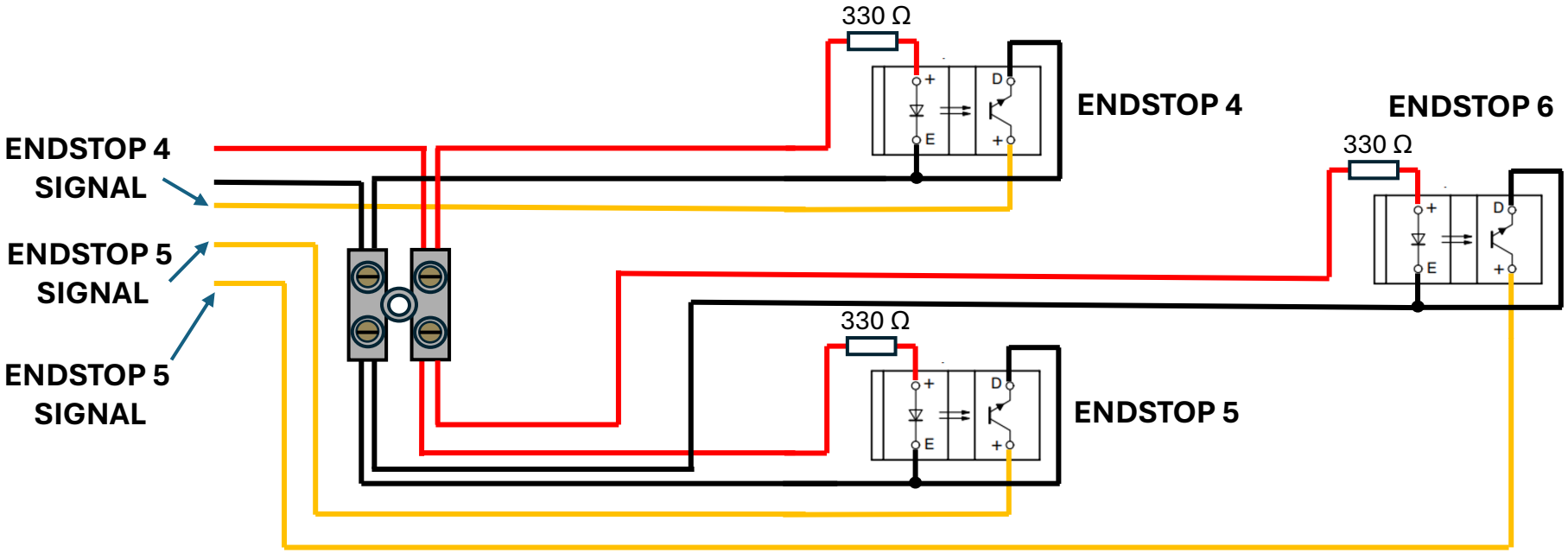
8x M3x30
8x M3 nut

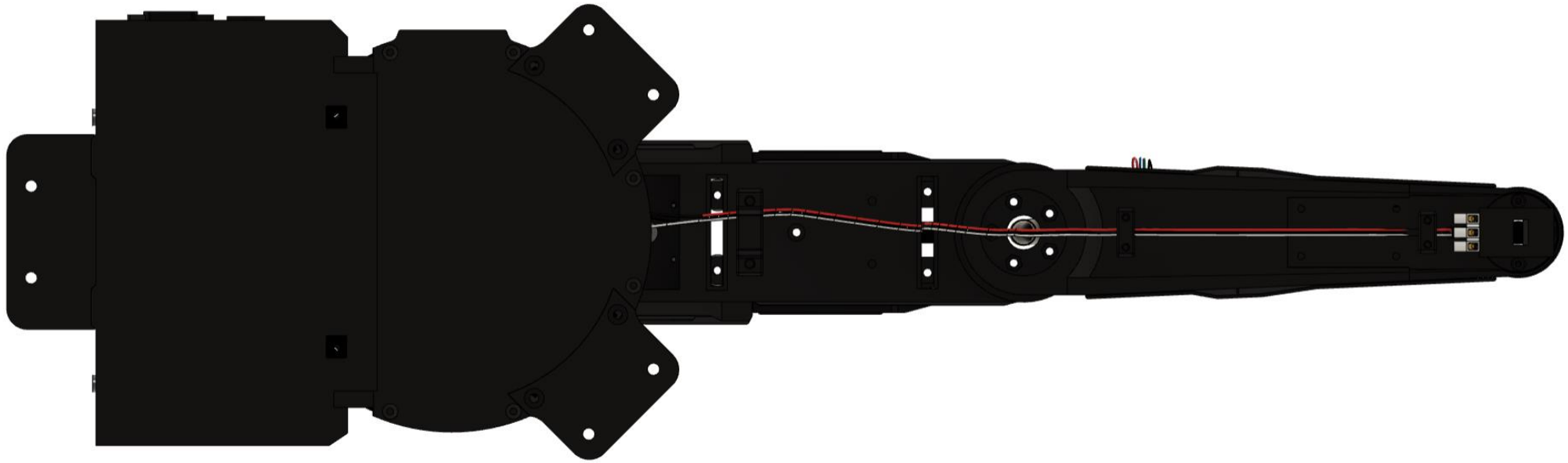




Now the three endstops 4,5,6 can be connected. The connection of endstop 4 and endstop 5 has been showed before. Endstop 6 will be connected to the same point.

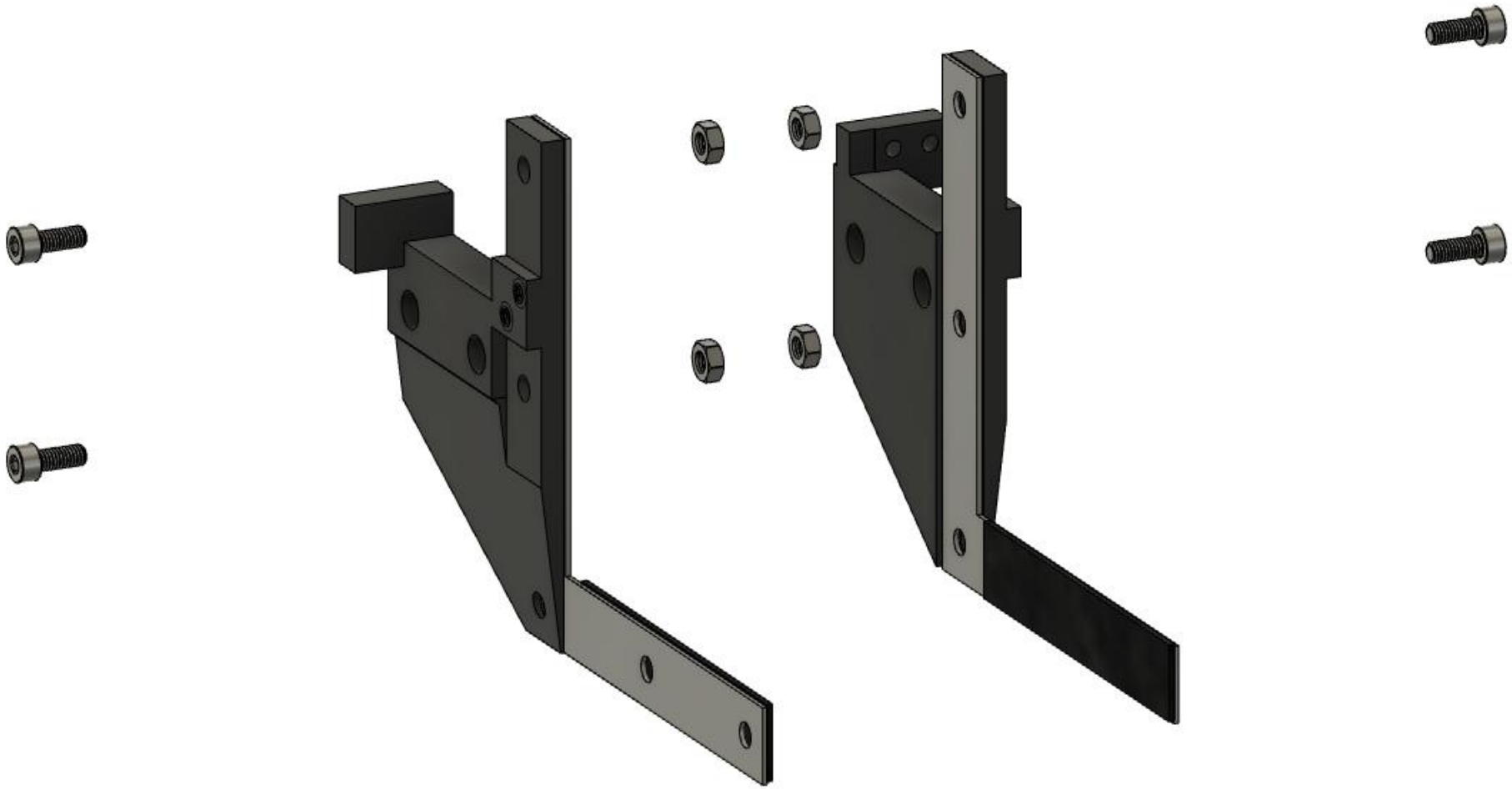

GND
5V
ENDSTOP 4
ENDSTOP 6
ENDSTOP 5


330 Ω
ENDSTOP 4
ENDSTOP 6
330 Ω
ENDSTOP 4
SIGNAL
ENDSTOP 5
SIGNAL
ENDSTOP 5
SIGNAL
330 Ω
ENDSTOP 5
D
E

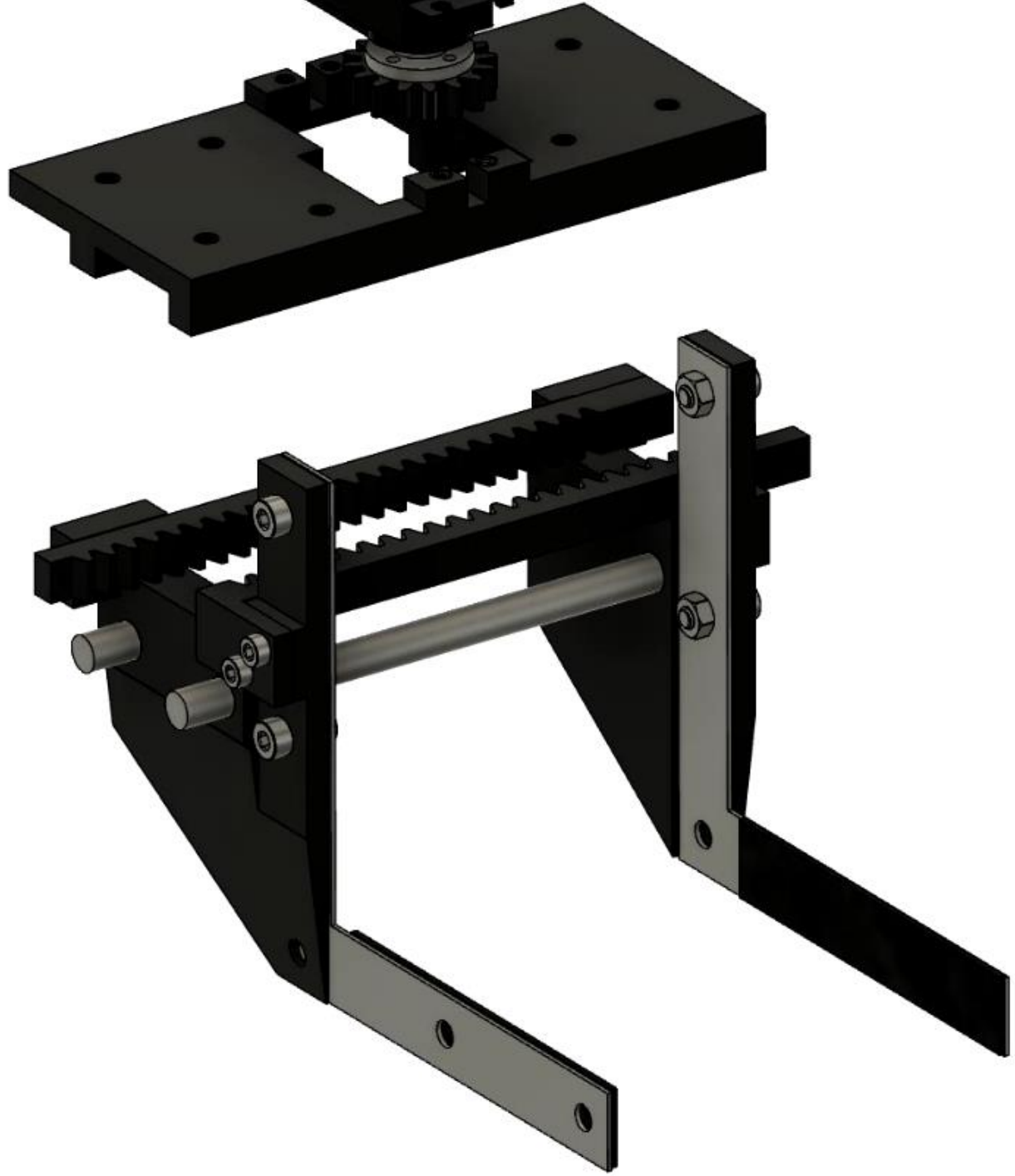


View from underneath

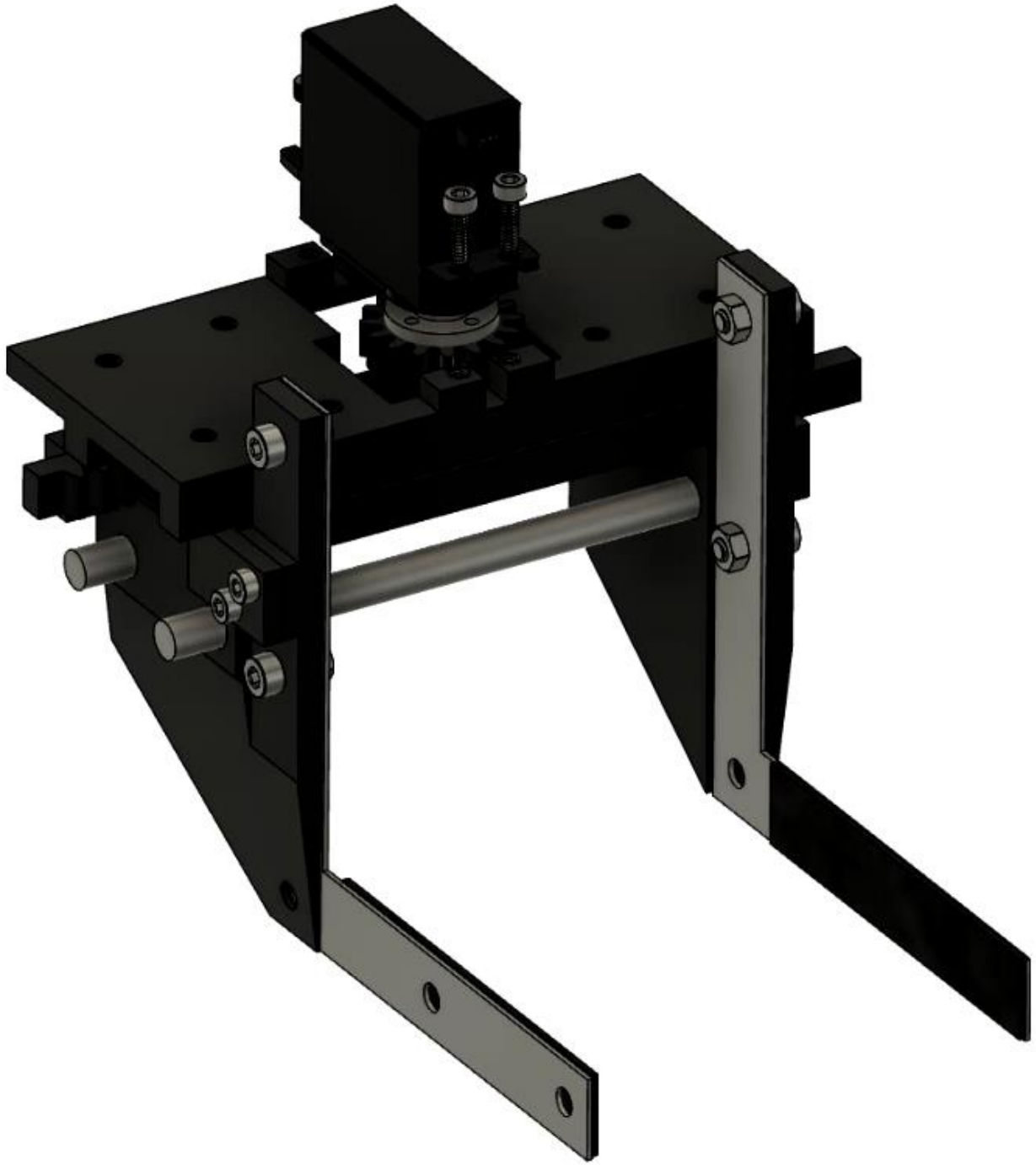
4x M4x10
4x M4 nut



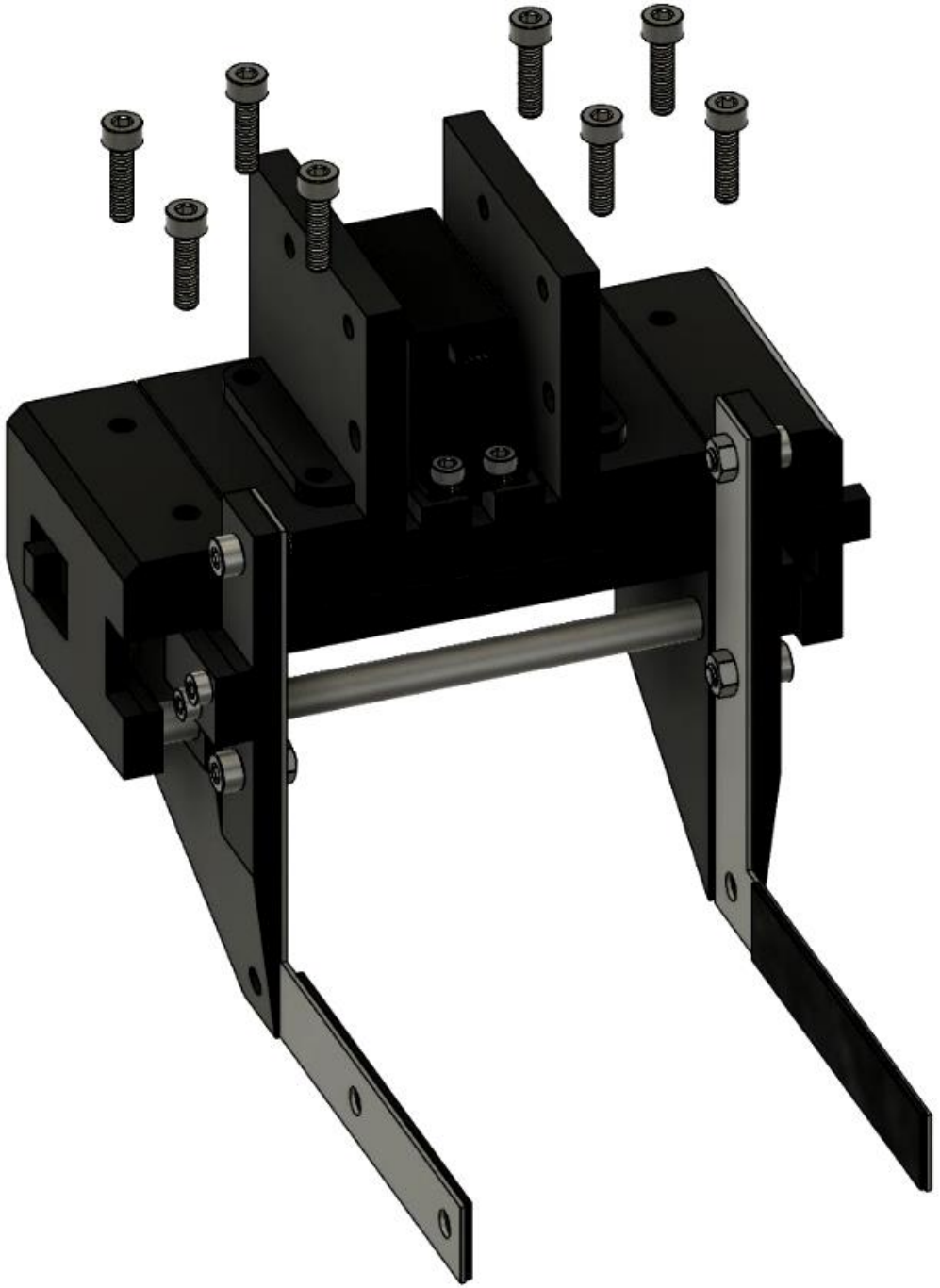
4x M3x10



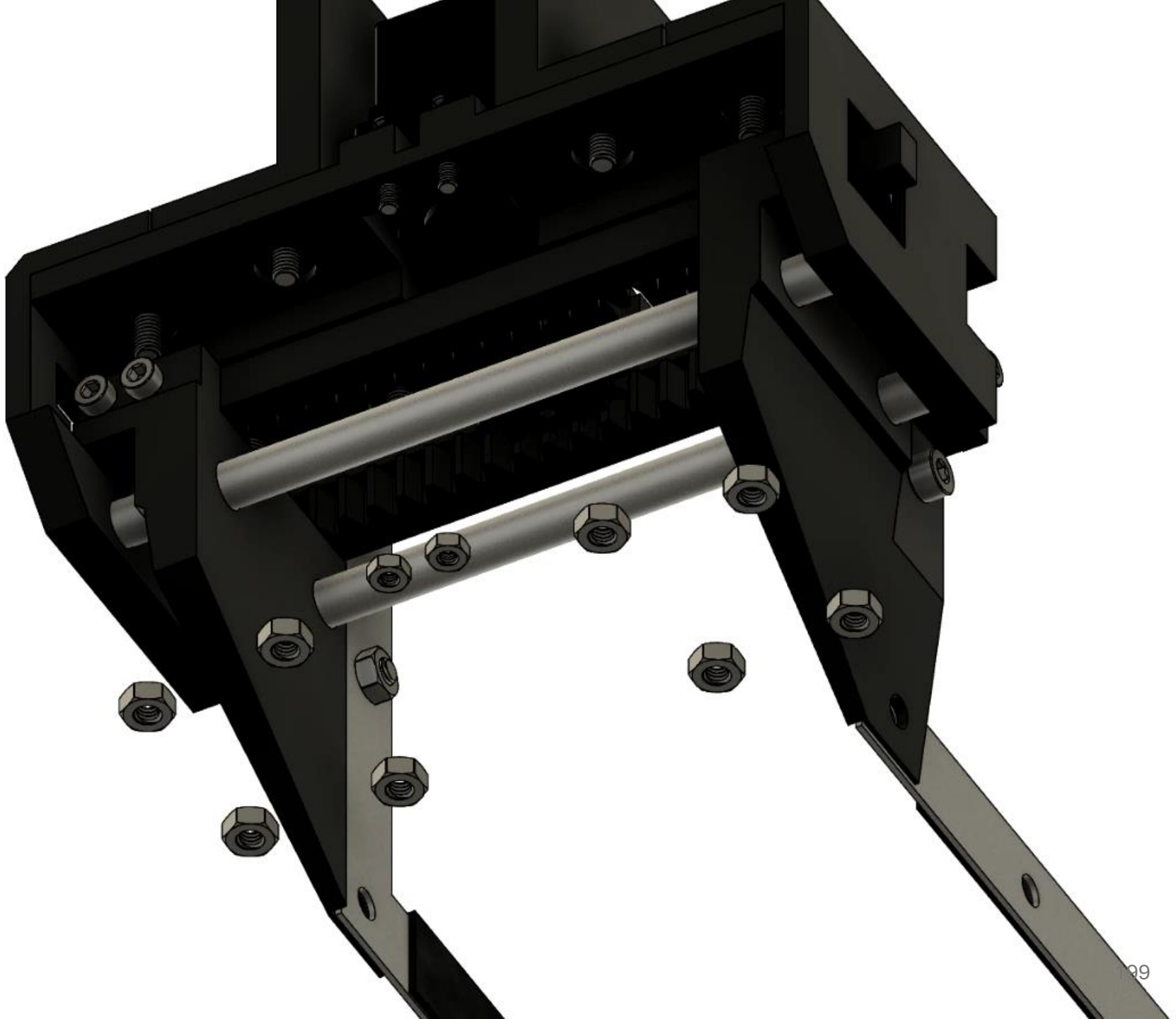
1x DS3218 Servo
2x M3x10
2x M3x15
2x M3 nut



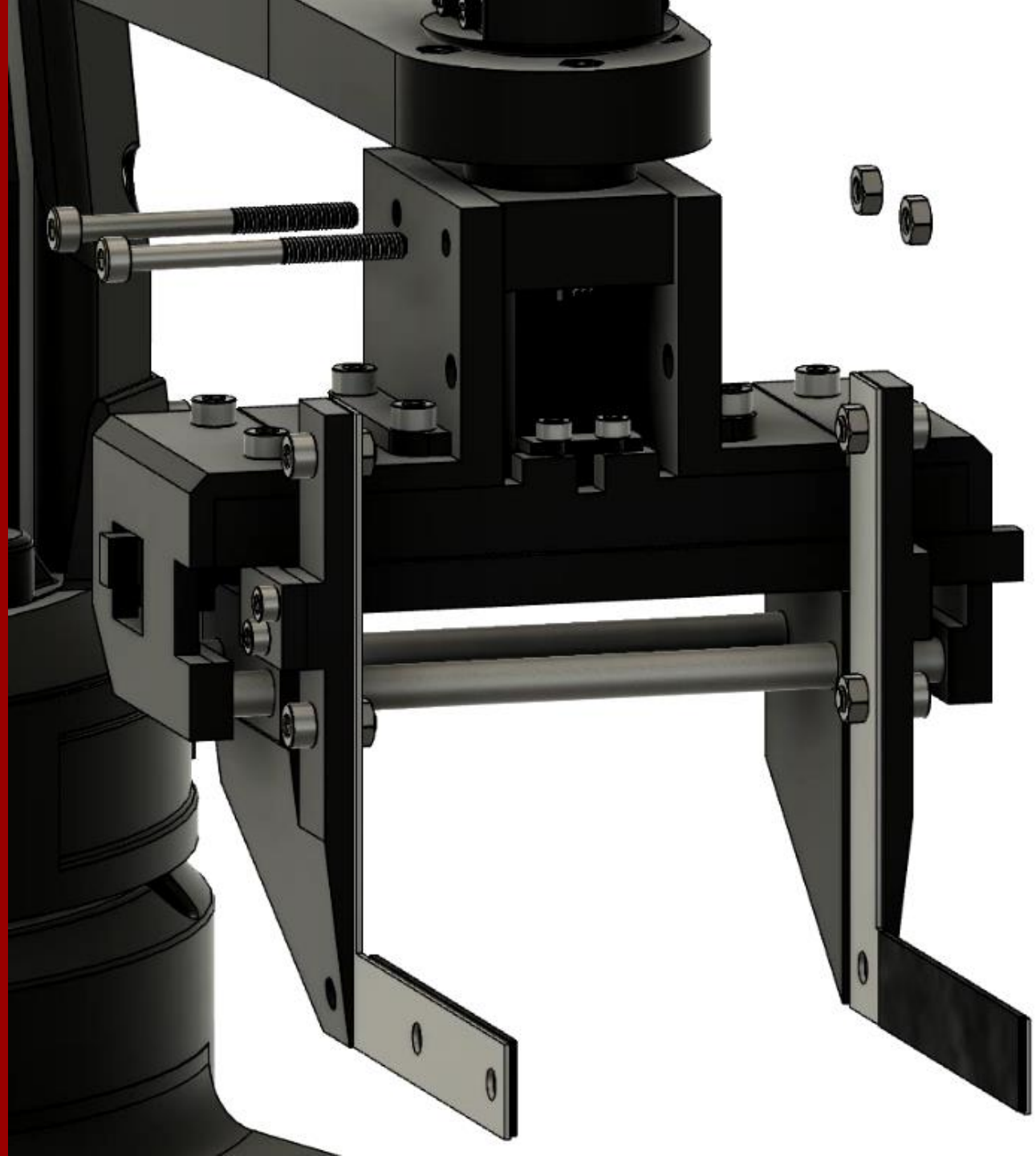
8x M4x15

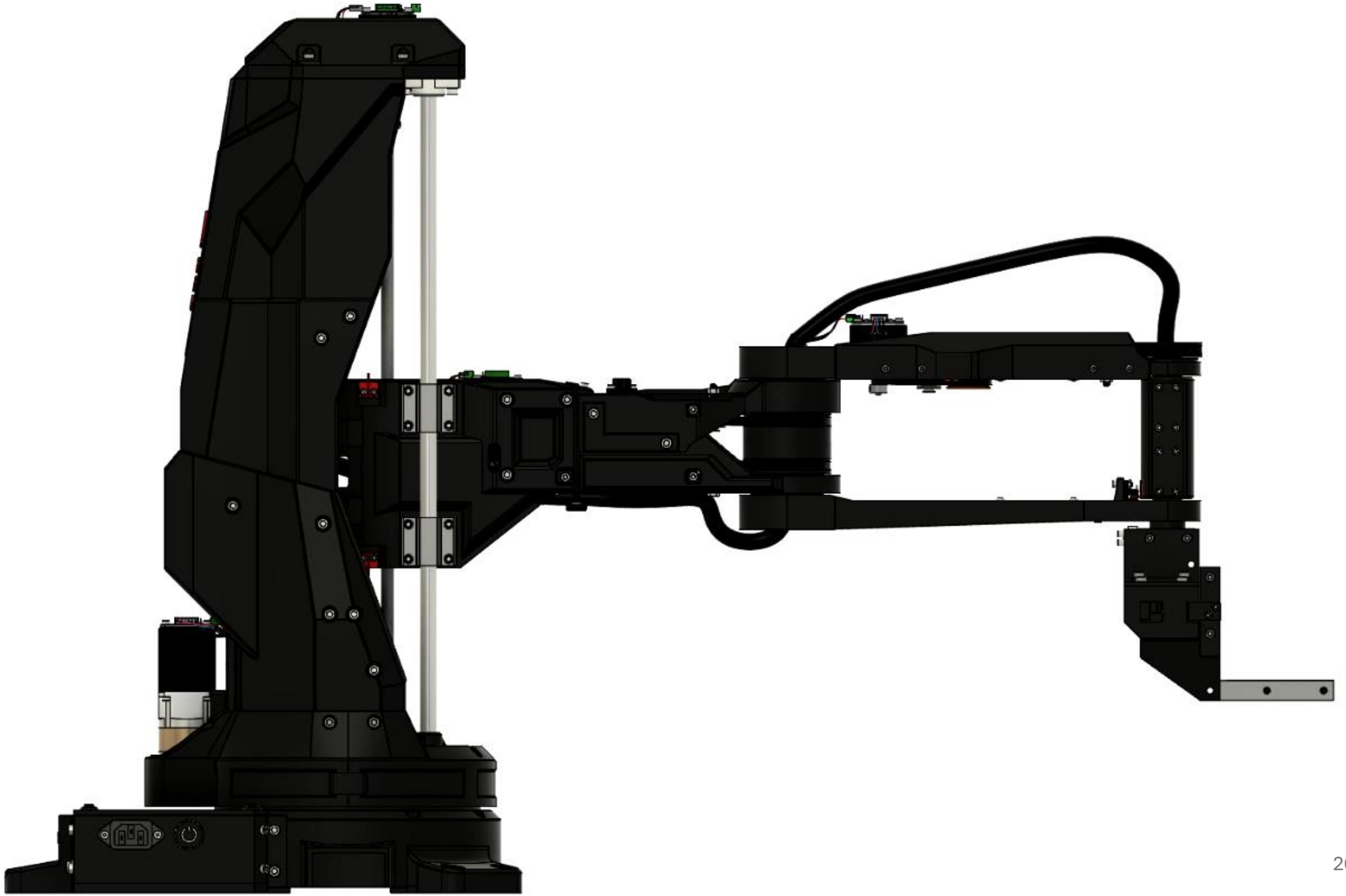


8x M4 nut
2x M3 nut



2x M5x45
2x M5 nut





ELECTRONICS

Actuators:

1x Nema 17 17HE12-1204S 

2x Nema 17 17HS19-2004S1 

1x Nema 17 17HS15-1504S1 

1x DS3218 Servo 

Control boards:

4x MKS Servo42C 

1x MKS Gen L2.1 

1x Raspberry Pi 4 

Sensors:

6x TCST2103 or MOC70T3  

Power:

1x 400W Power Supply 

1x 24V/5V Step Down 

1x Power Button 

1x C14 PLUG 

1x USB C Wire 

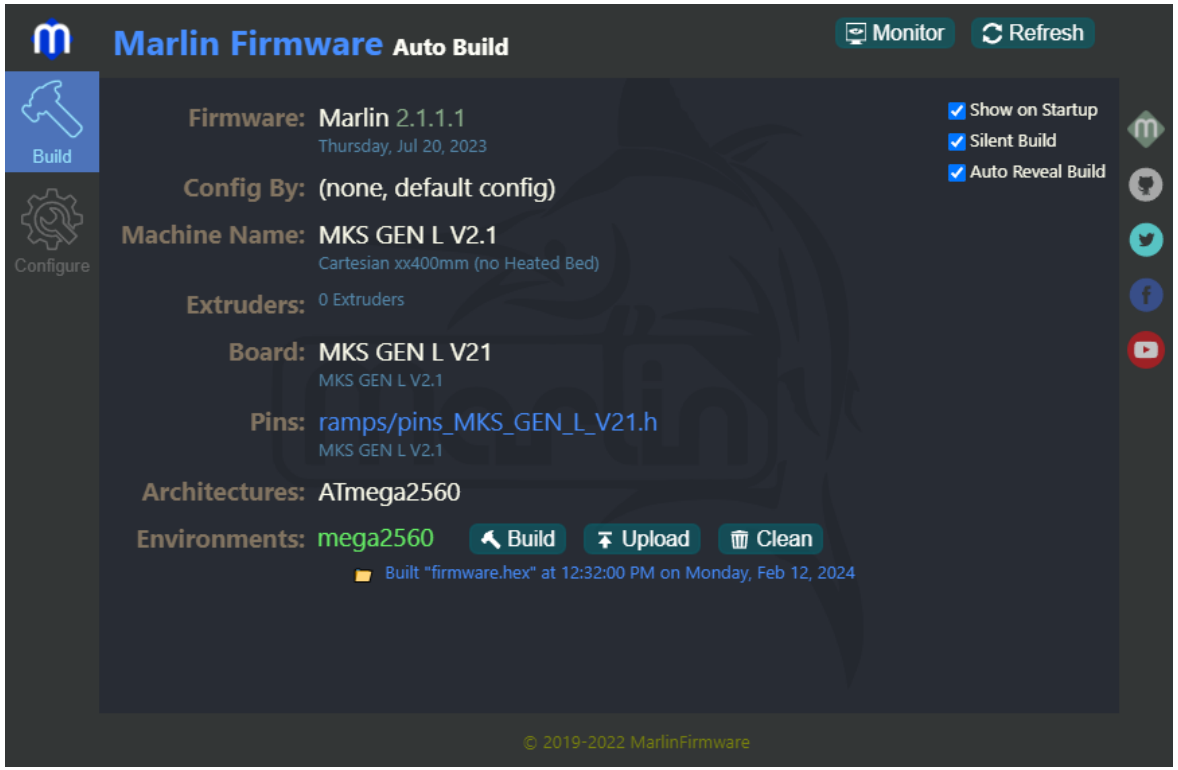
Miscellaneous:

1x TXS0108E 

MKS GEN L2.1 SETUP

SETUP

1. Download the changed software
2. Add the extension “Auto Build Marlin” to Visual Studio Code
   https://marlinfw.org/docs/basics/auto_build_marlin.html
   https://marlinfw.org/docs/basics/install_platformio_vscode.html
3. Build the software
4. Upload the software to the MKS Gen L2.1



BRIEF SUMMARY OF MARLIN CHANGES

Since the software for the motherboard has been designed with 3d printers and CNC machines in mind, some necessary changes must be made in order to use it for this application.

Marlin is meant to be customized for specific applications through two configuration files:

- Configuration.h
- Configuration_adv.h

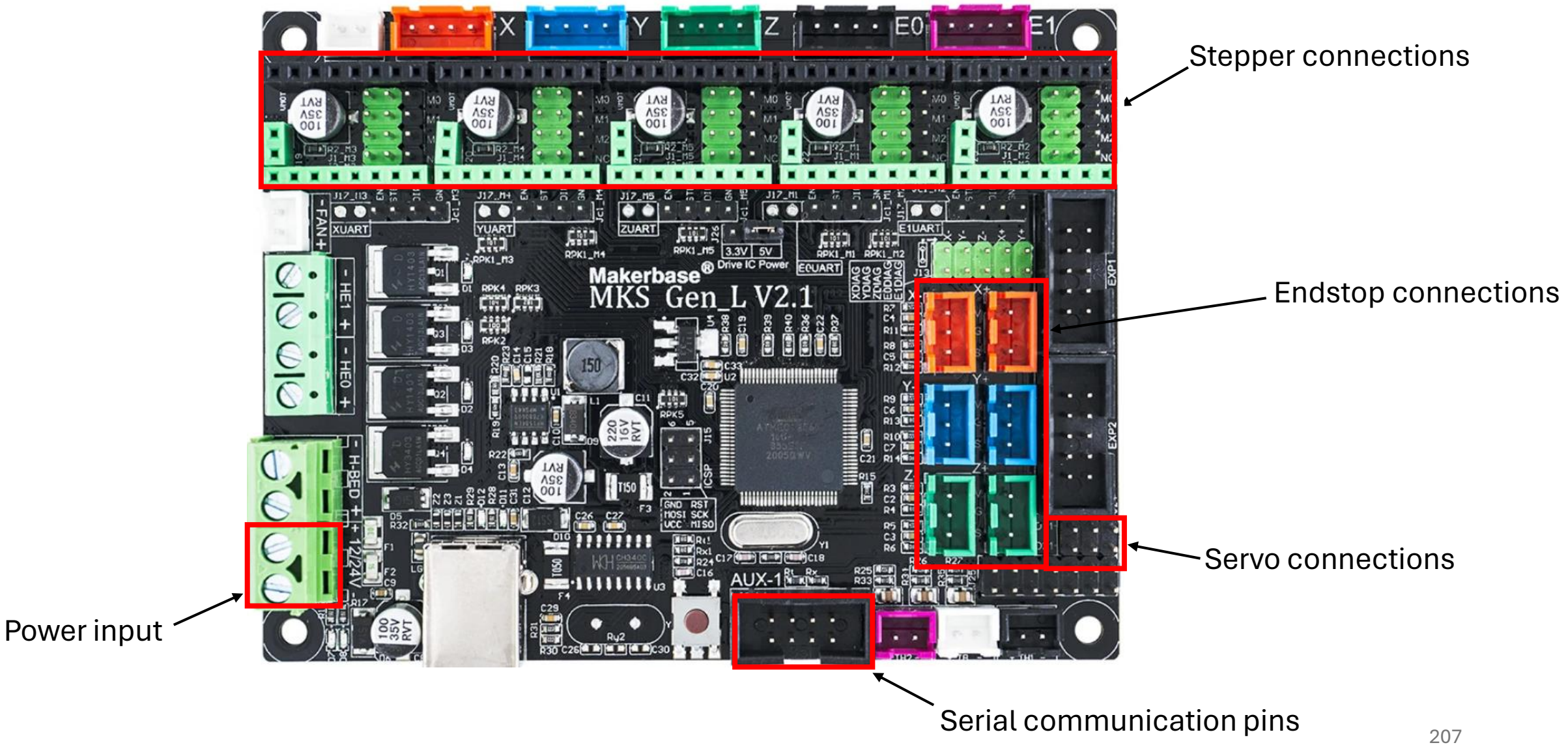
In these scripts, it is important to define the motherboard model to match the pinout, define an additional axis (3 is the default so an additional has to be added since 4 are required), define the motion parameters (microstepping, speed, acceleration) define endstops and their logic, disable software endstops.

Additionally, some custom changes had to be made such as disabling thermal protection settings, disabling the motion interruption for joint 1 and joint 4 (since these joints should are stopped only during homing sequences). These are only some of the more important changes. It is possible to disable more of the unused features however it must be noted that there are multiple scripts depending on each other and making excessive changes might result in the script not working in the end.

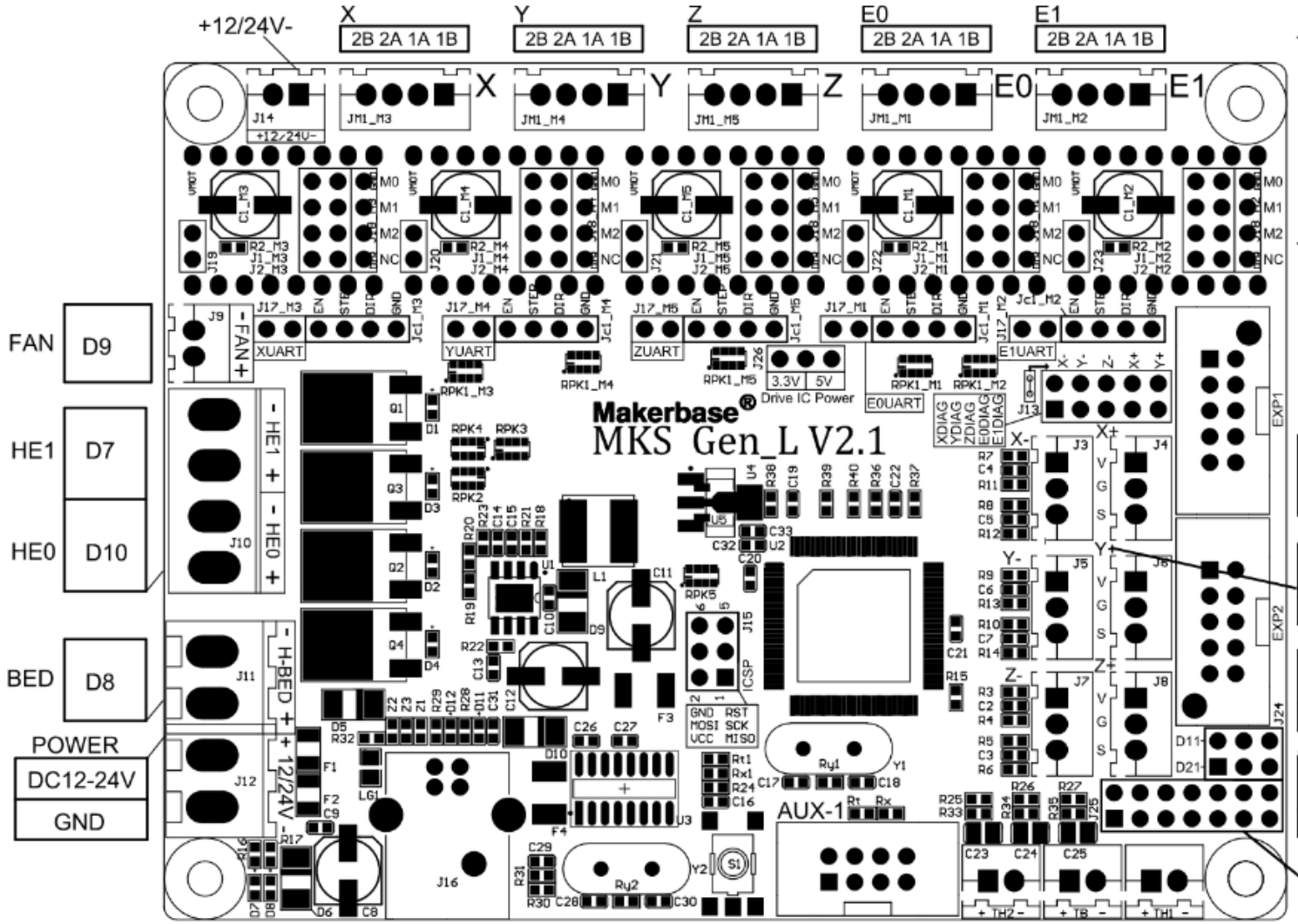
A folder with the changed Marlin can be found on Github.

Additionally in Visual Studio Code there is an extension called “Auto Build Marlin” it can be used to compile all the scripts and then upload it to the motherboard.

MKS GEN L2.1 PORTS



MKS GEN L2.1 PINOUT



TMC2208-UART

TX	RX	
D40	A9	X
A5	A10	Y
D42	A11	Z
D44	A12	E0
D20	D12	E1

EN	STEP	DIR		
D38	A0	A1	GND	X
A2	A6	A7	GND	Y
A8	D46	D48	GND	Z
D24	D26	D28	GND	E0
D30	D36	D34	GND	E1

TMC2130-SPI

MOSI	SCLK	CS	MISO	
D51	D52	A9	D50	X
		A10		Y
		A11		Z
		A12		E
		D12		E

XDIAG	YDIAG	ZDIAG	E0DIAG	E1DIAG
X-	Y-	Z-	X+	Y+
D3	D14	D18	D2	D15

X-	X+
5V	5V
GND	GND
D3	D2

Y-	Y+
5V	5V
GND	GND
D14	D15

Z-	Z+
5V	5V
GND	GND
D18	D19

EXP1

D37	D35
D17	D16
D23	D25
D27	D29
GND	5V

EXP2

D50	D52
D31	D53
D33	D51
D49	RST
GND	NC

AUX-1

5V	GND	TX	RX
5V	GND	NC	NC

THM

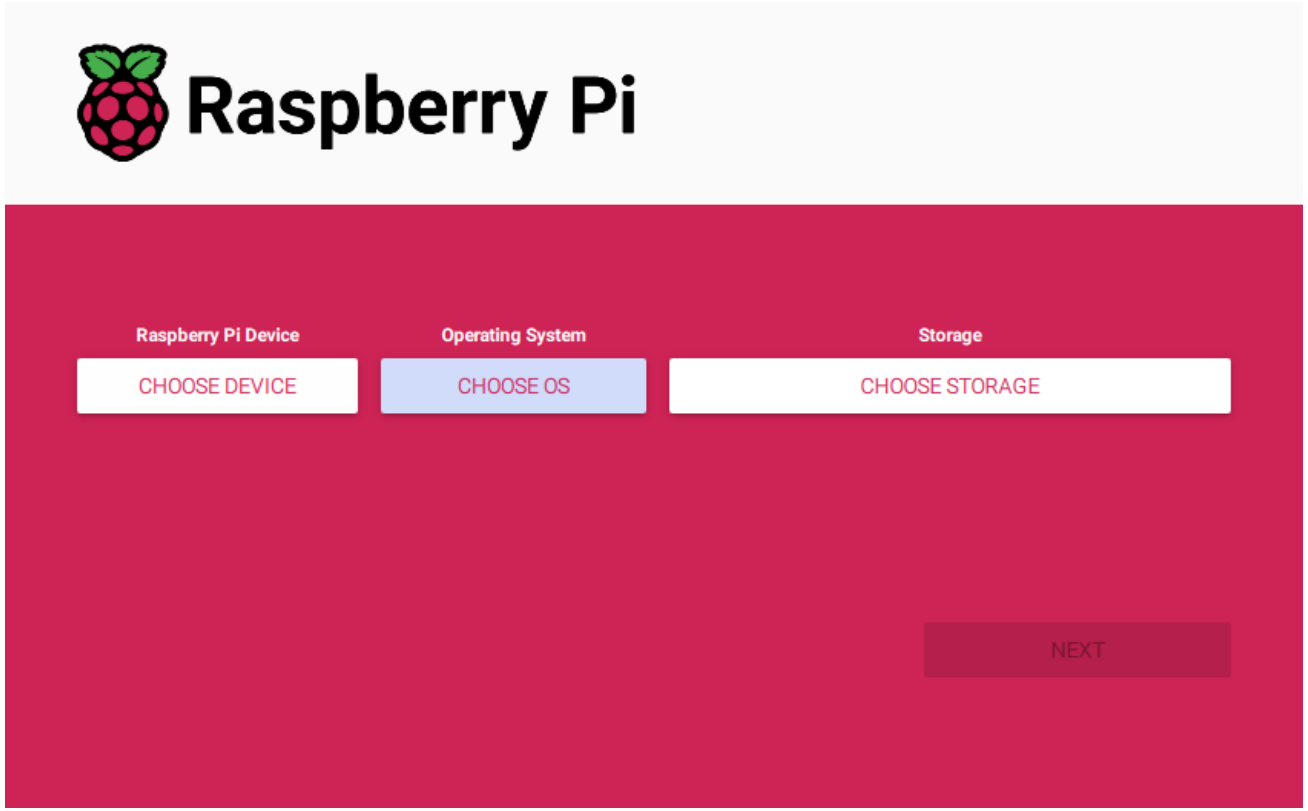
TH2	TB	TH1
A15	A14	A13

A4	A3	D6	D45	D11	5V	GND
D4	D5	D43	D47	D21	5V	GND
				D39	5V	GND
				D32	5V	GND

RASPBERRY SETUP

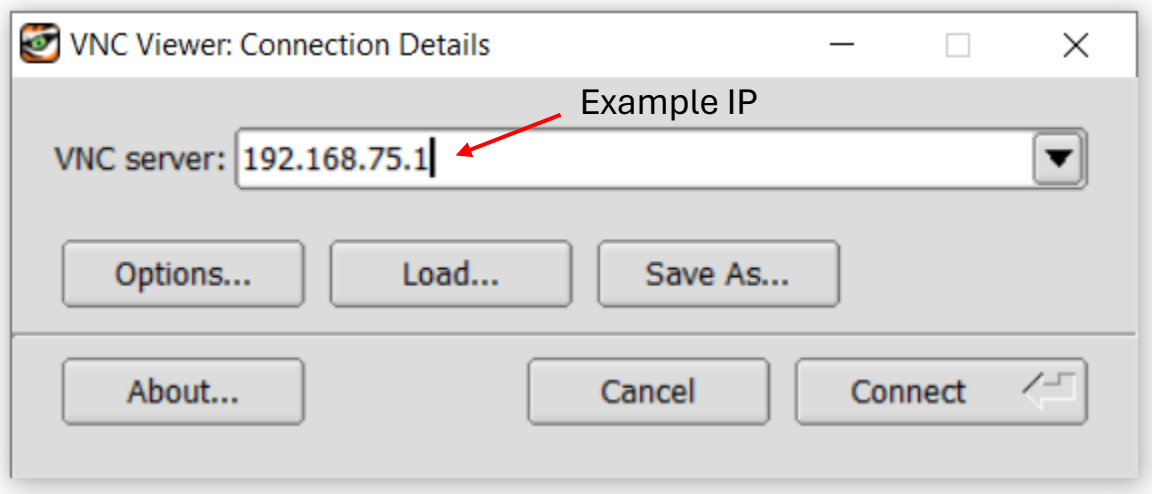
SETUP

1. Download and install Raspberry Pi Imager
2. Choose device
3. Choose storage (Recommended at least 64GB of SD card storage)
4. Choose OS (Recommended Raspberry Pi OS 64-bit by default)
5. Configure WiFi network for VNC (to use as a remote desktop)
6. Transfer the python scripts onto the Raspberry



VNC – temporary solution

It is advised to verify if the VNC has been set up correctly. To do so, download Tiger VNC and try to connect to the Raspberry Pi. The Pi's IP address is required (the device should be visible with the name "scara").



VNC is a temporary solution which serves as a wireless display.
In future applications the robot scripts could be called from a remote desktop or a local machine instead of manually running them using VNC.

MKS SERVO42C

SETUP

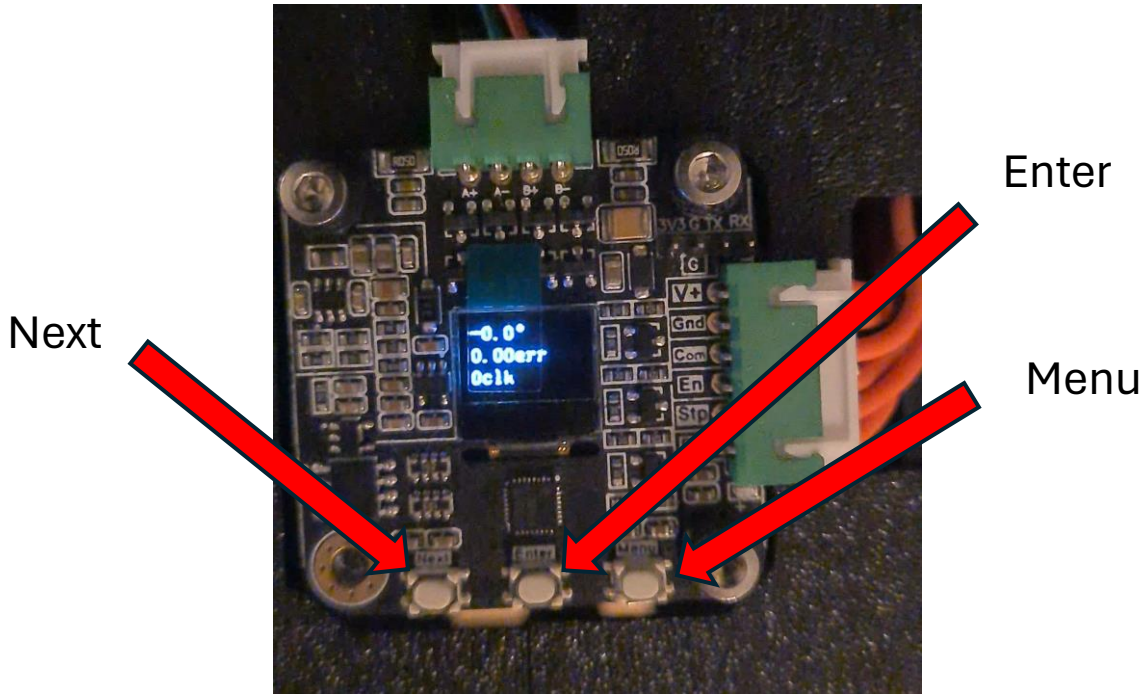
Navigate through the settings by using the OLED display and buttons

1. Go to “Mode” and select CR_vFOC
2. Go to “MotType” and select 1.8
3. Go to “MStep” and select “32”

4. Stall protection (optional)

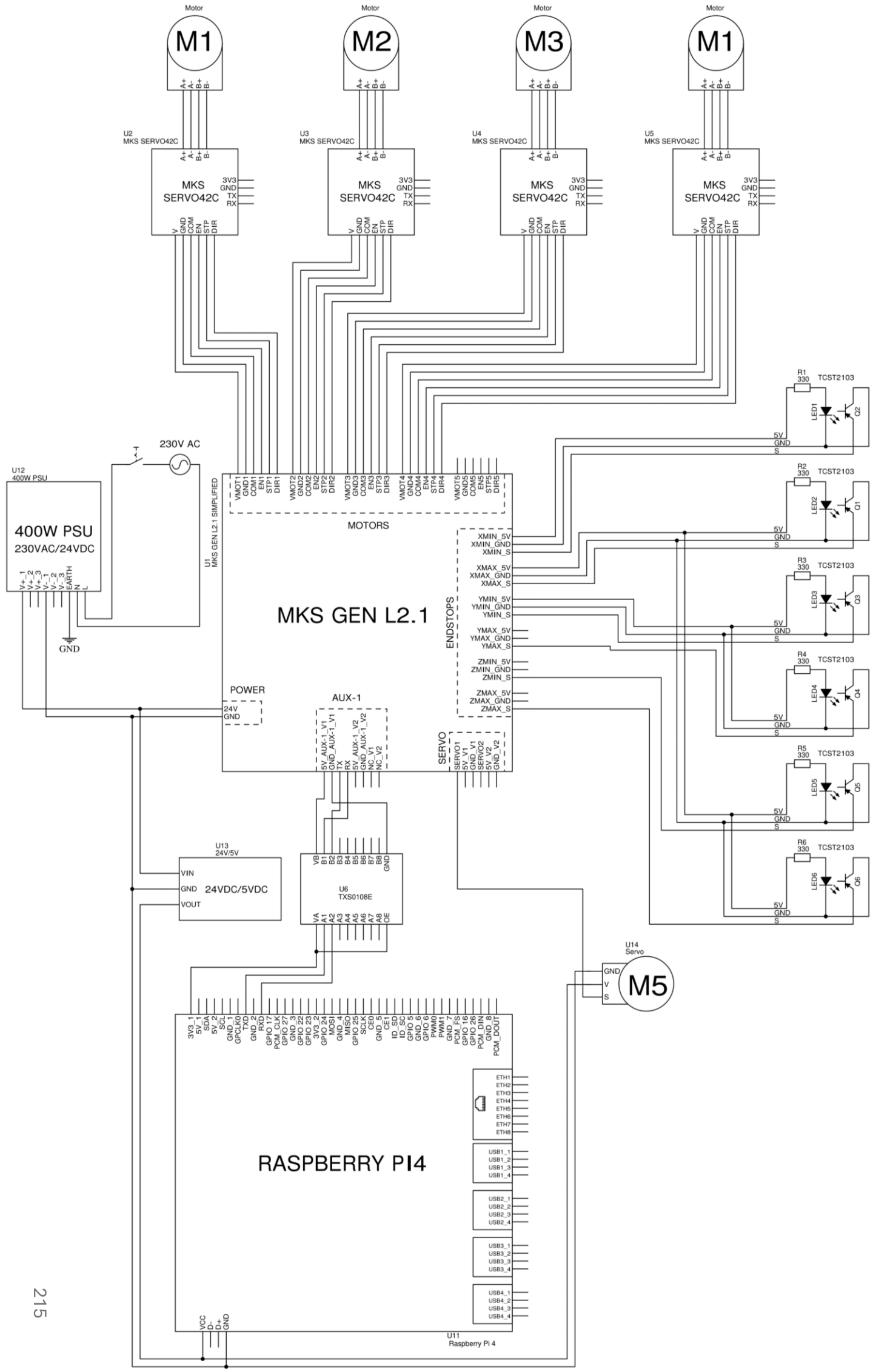
Go to “Protect” and select “Enable” (This feature is quite sensitive and can result in the motor being disabled accidentally while performing a motion. The triggered state is indicated by a blue LED).

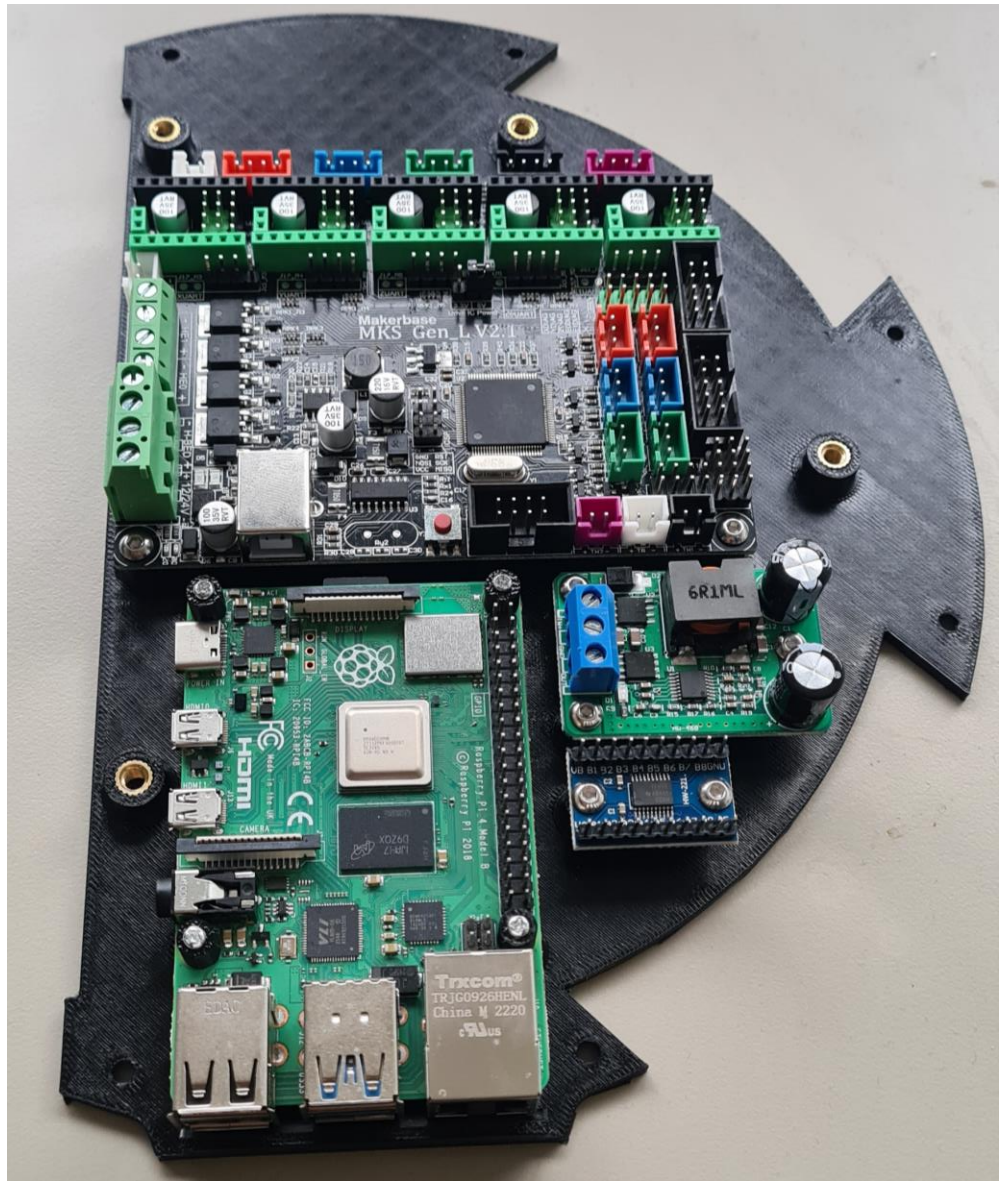
5. Go to “Cal” and let the motor calibrate (by slowly moving back and forth).

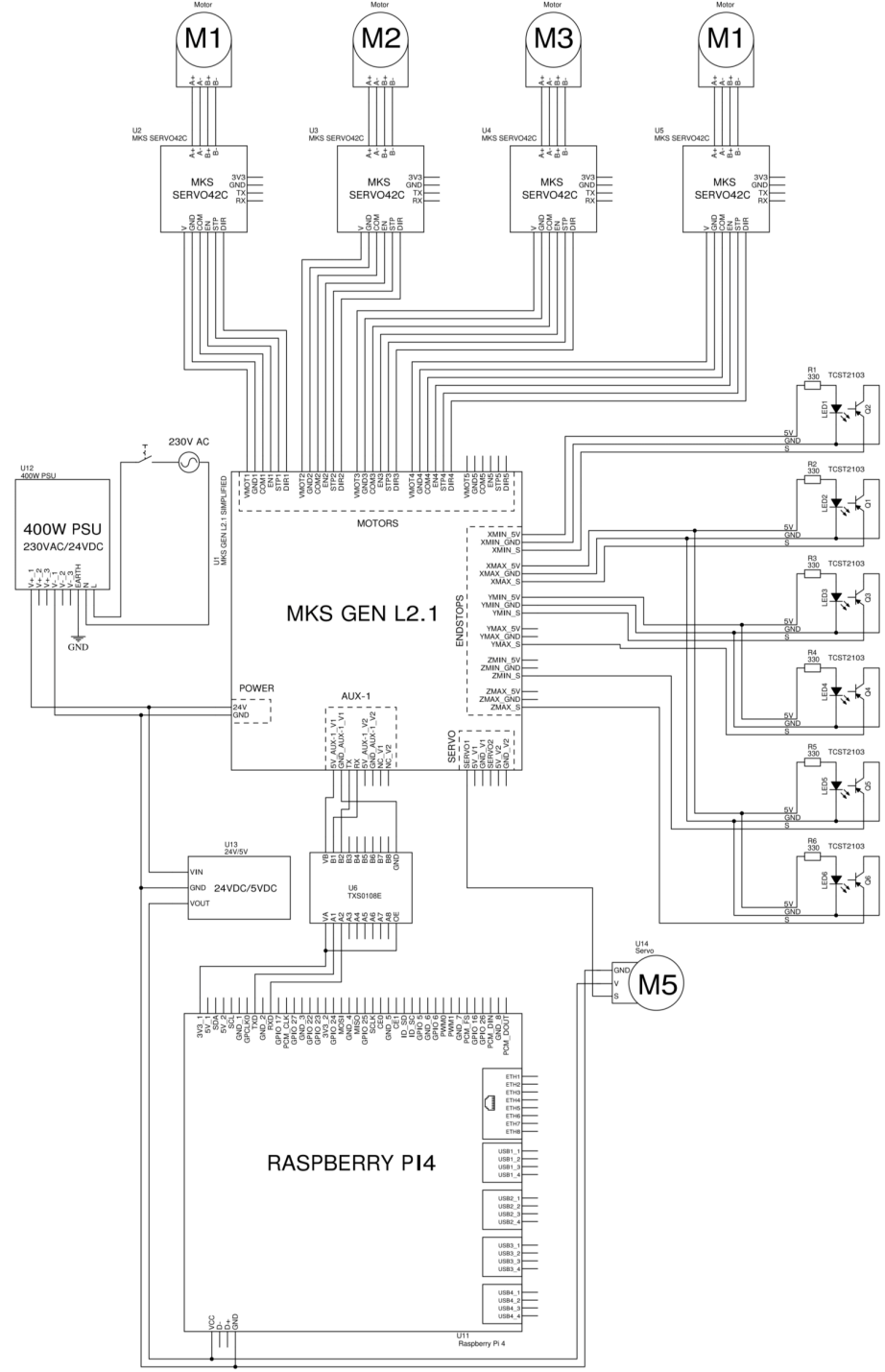


NOTE:
Washers might be required for mounting the MKS Servo.

SCHEMATIC


Motor
M1
M2
M3
M1
A+
A-
B+
B-
U2
MKS SERVO42C
U3
MKS SERVO42C
U4
MKS SERVO42C
U5
MKS SERVO42C
MKS SERVO42C
3V3
GND
TX
RX
V
GND
COM
EN
STP
DIR
230V AC
U12
400W PSU
400W PSU
230VAC/24VDC
V+_1
V+_2
V+_3
V-_1
V-_2
V-_3
EARTH
N
L
GND
U1
MKS GEN L2.1 SIMPLIFIED
VMOT1
GND1
COM1
EN1
STP1
DIR1
VMOT2
GND2
COM2
EN2
STP2
DIR2
VMOT3
GND3
COM3
EN3
STP3
DIR3
VMOT4
GND4
COM4
EN4
STP4
DIR4
VMOT5
GND5
COM5
EN5
STP5
DIR5
MOTORS
MKS GEN L2.1
XMIN_5V
XMIN_GND
XMIN_S
XMAX_5V
XMAX_GND
XMAX_S
YMIN_5V
YMIN_GND
YMIN_S
YMAX_5V
YMAX_GND
YMAX_S
ZMIN_5V
ZMIN_GND
ZMIN_S
ZMAX_5V
ZMAX_GND
ZMAX_S
ENDSTOPS
POWER
24V
GND
AUX-1
5V_AUX-1_V1
GND_AUX-1_V1
TX
RX
5V_AUX-1_V2
GND_AUX-1_V2
NC_V1
NC_V2
SERVO
SERVO1
5V_V1
GND_V1
SERVO2
5V_V2
GND_V2
R1
330
TCST2103
LED1
Q2
R2
LED2
Q1
R3
LED3
Q3
R4
LED4
Q4
R5
LED5
Q5
R6
LED6
Q6
5V
GND
S
U13
24V/5V
VIN
GND
VOUT
24VDC/5VDC
U6
TXS0108E
VB
B1
B2
B3
B4
B5
B6
B7
B8
GND
VA
A1
A2
A3
A4
A5
A6
A7
A8
OE
U14
Servo
M5
GND
V
S
3V3_1
5V_1
SDA
SCL
5V_2
GND_1
GPCLK0
TXD
GND_2
RXD
GPIO 17
PCM_CLK
GPIO 27
GND_3
GPIO 22
GPIO 23
3V3_2
GPIO 24
MOSI
GND_4
MISO
GPIO 25
SCLK
CE0
GND_5
CE1
ID_SD
ID_SC
GPIO 5
GND_6
GPIO 6
PWM0
PWM1
GND_7
PCM_FS
GPIO 16
GPIO 26
PCM_DIN
GND_8
PCM_DOUT
RASPBERRY PI4
ETH1
ETH2
ETH3
ETH4
ETH5
ETH6
ETH7
ETH8
USB1_1
USB1_2
USB1_3
USB1_4
USB2_1
USB2_2
USB2_3
USB2_4
USB3_1
USB3_2
USB3_3
USB3_4
USB4_1
USB4_2
USB4_3
USB4_4
VCC
D-
D+
GND
U11
Raspberry Pi 4

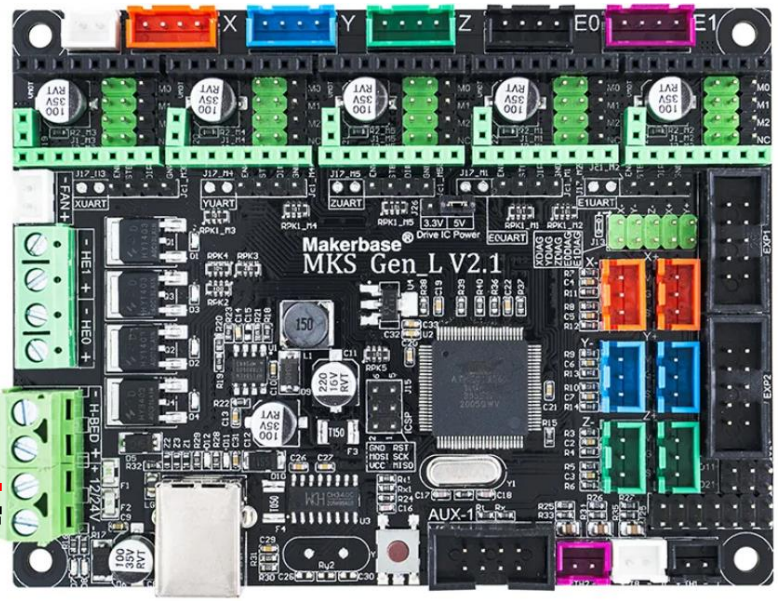

Makerbase
MKS Gen_L V2.1
6R1ML
Trxcom®
TRJG0926HENL
China 2220


M1
M2
M3
M1
MKS SERVO42C
230V AC
400W PSU
230VAC/24VDC
MKS GEN L2.1
MOTORS
ENDSTOPS
POWER
AUX-1
SERVO
24VDC/5VDC
TXS0108E
RASPBERRY PI4
M5
TCST2103

POWER CONNECTIONS

POWER SUPPLY

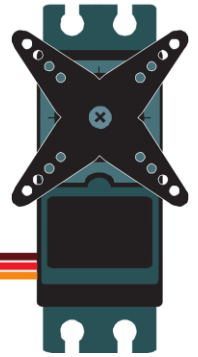
MKS GEN L2.1

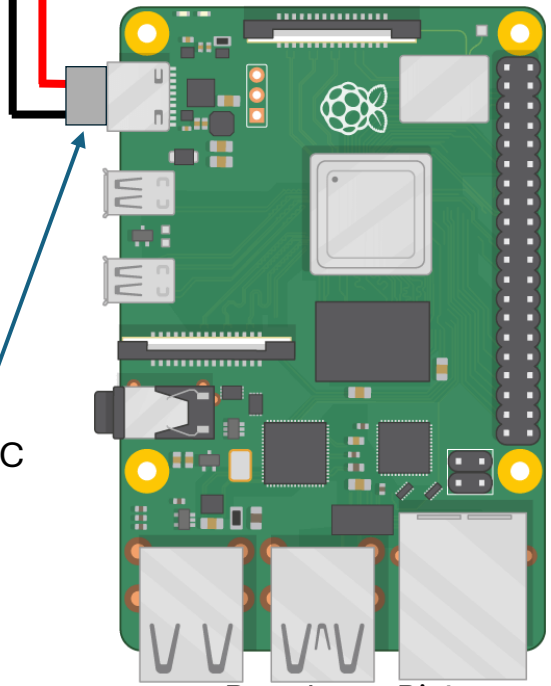


The power for the servo motor is **not supplied by the MKS motherboard**! The motherboard's servo pins have a low maximum current output which will likely be exceeded by the current drawn by the servo during operation.

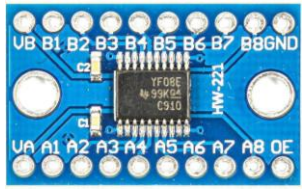
24V/5V step down

Servo





USB C

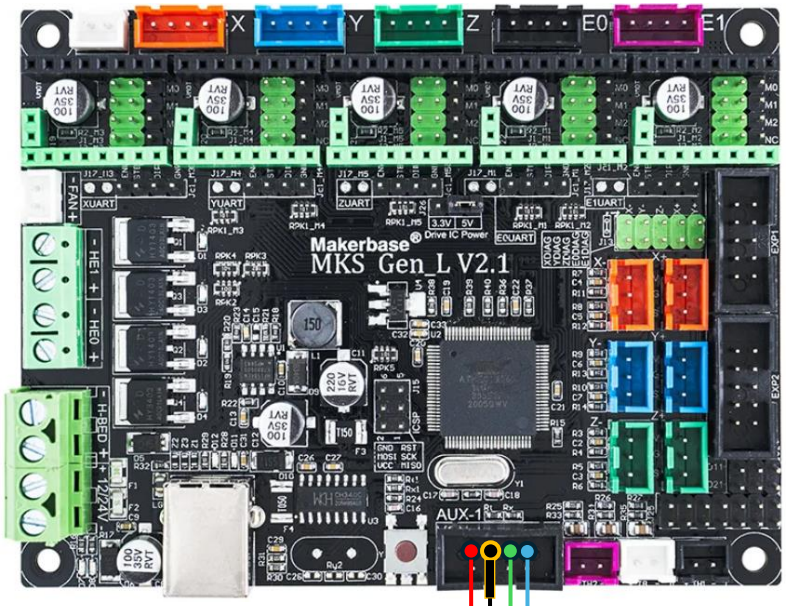


TXS0108E

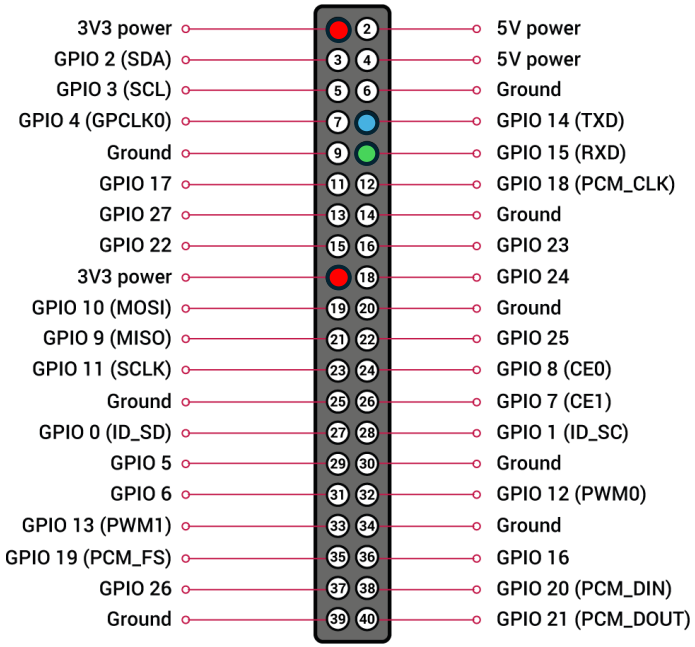
Note that the signal servo input should not be connected yet.

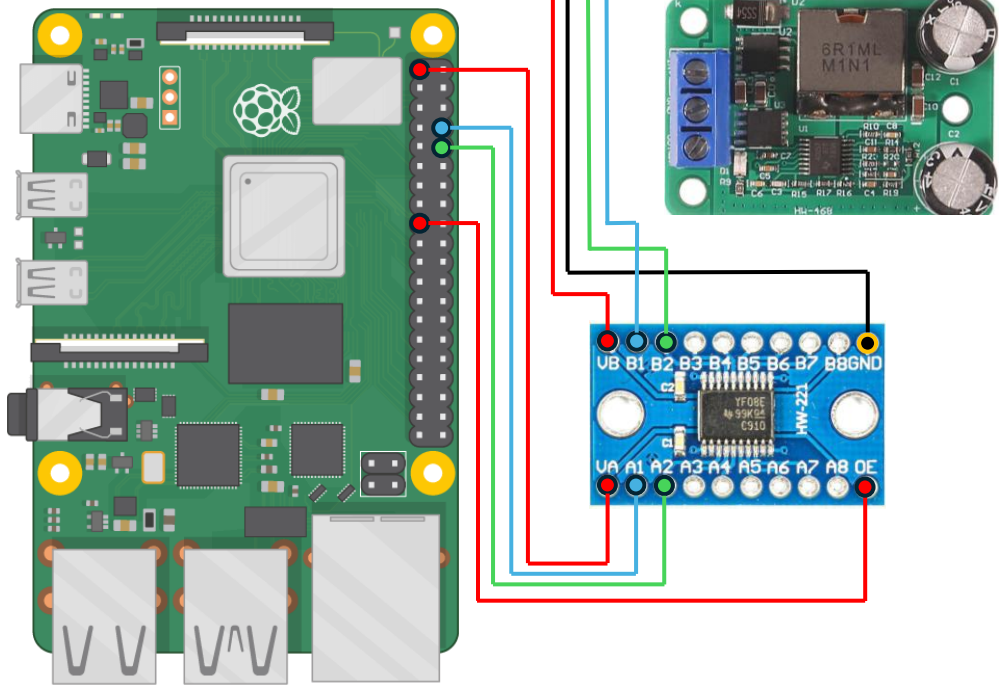
Raspberry Pi 4

LOGIC WIRING



A logic level converter is required to facilitate serial communication between 5V I/O ports (MKS board) and 3.3V I/O ports (Raspberry)

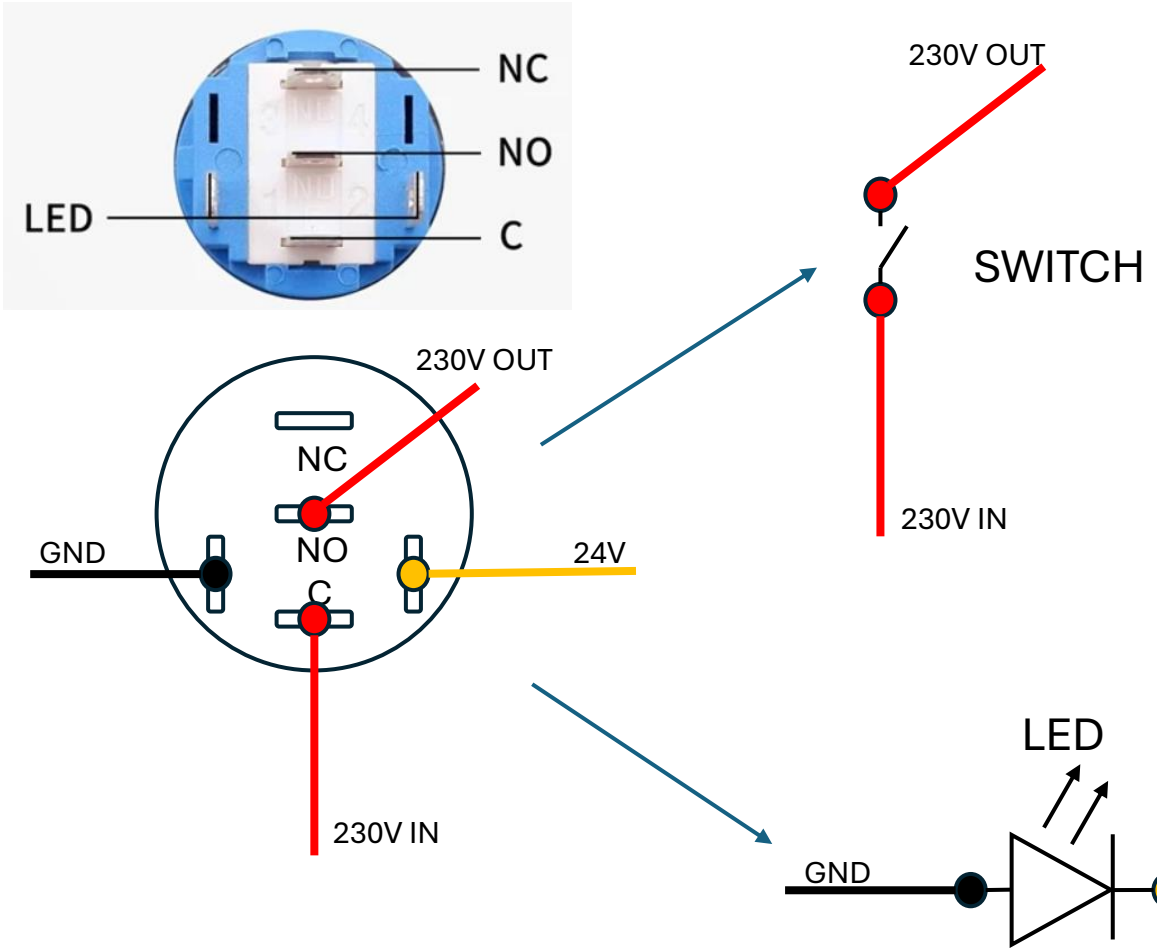




FRONT PANEL

The button and power socket need to be prepared for the front panel assembly.
The button consists of an LED and a switch. The button requires 4 wires and the power socket 3 wires

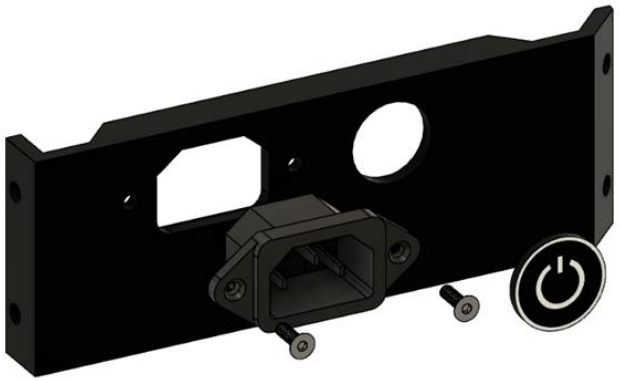
BUTTON WIRING



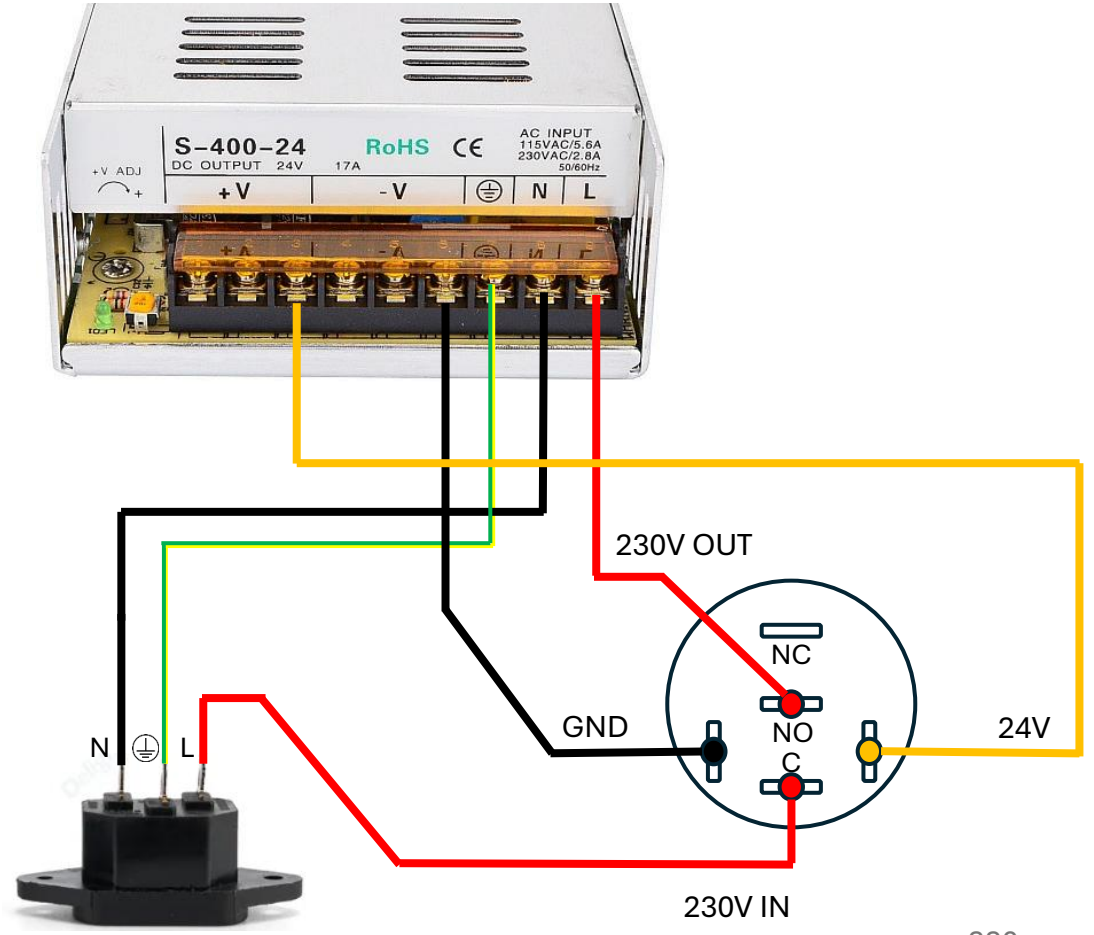
C14 POWER SOCKET WIRING

FRONT PANEL WIRING

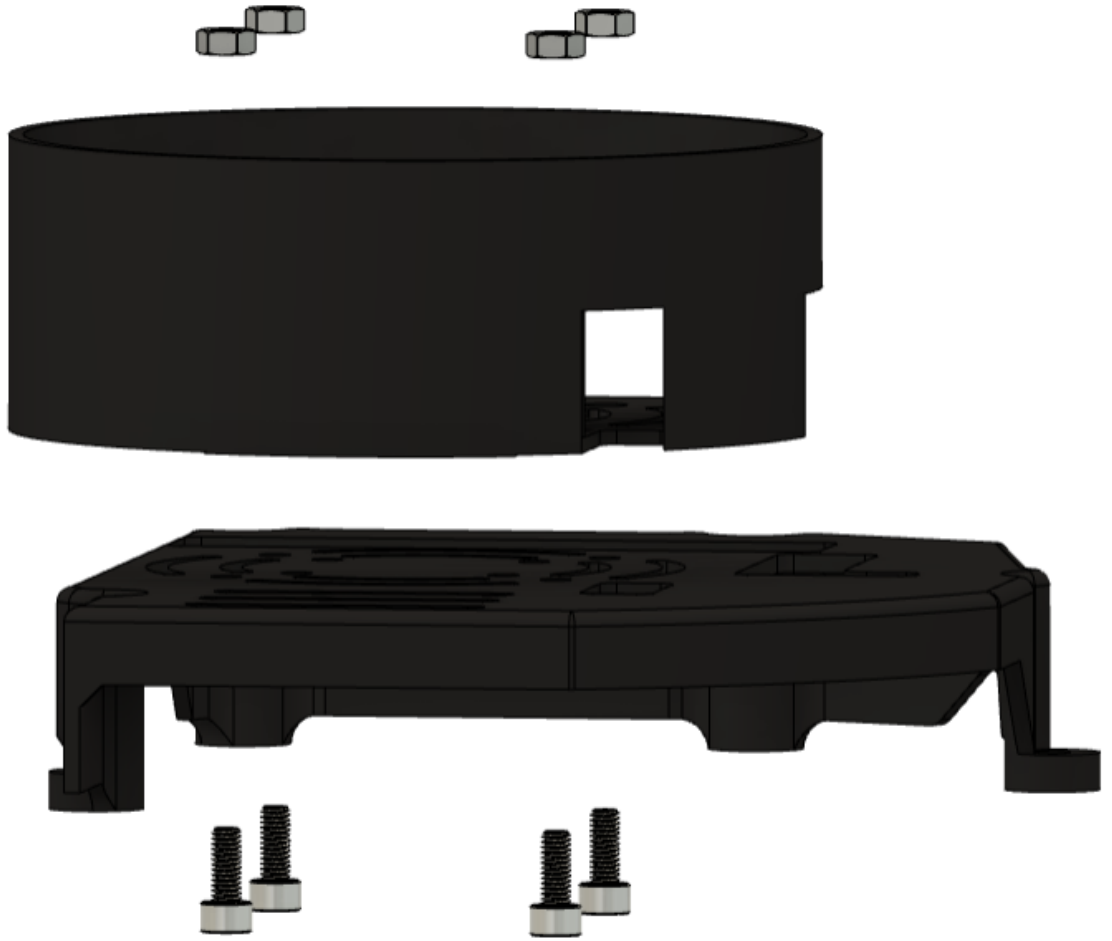
1. Mount the socket and button onto the front panel



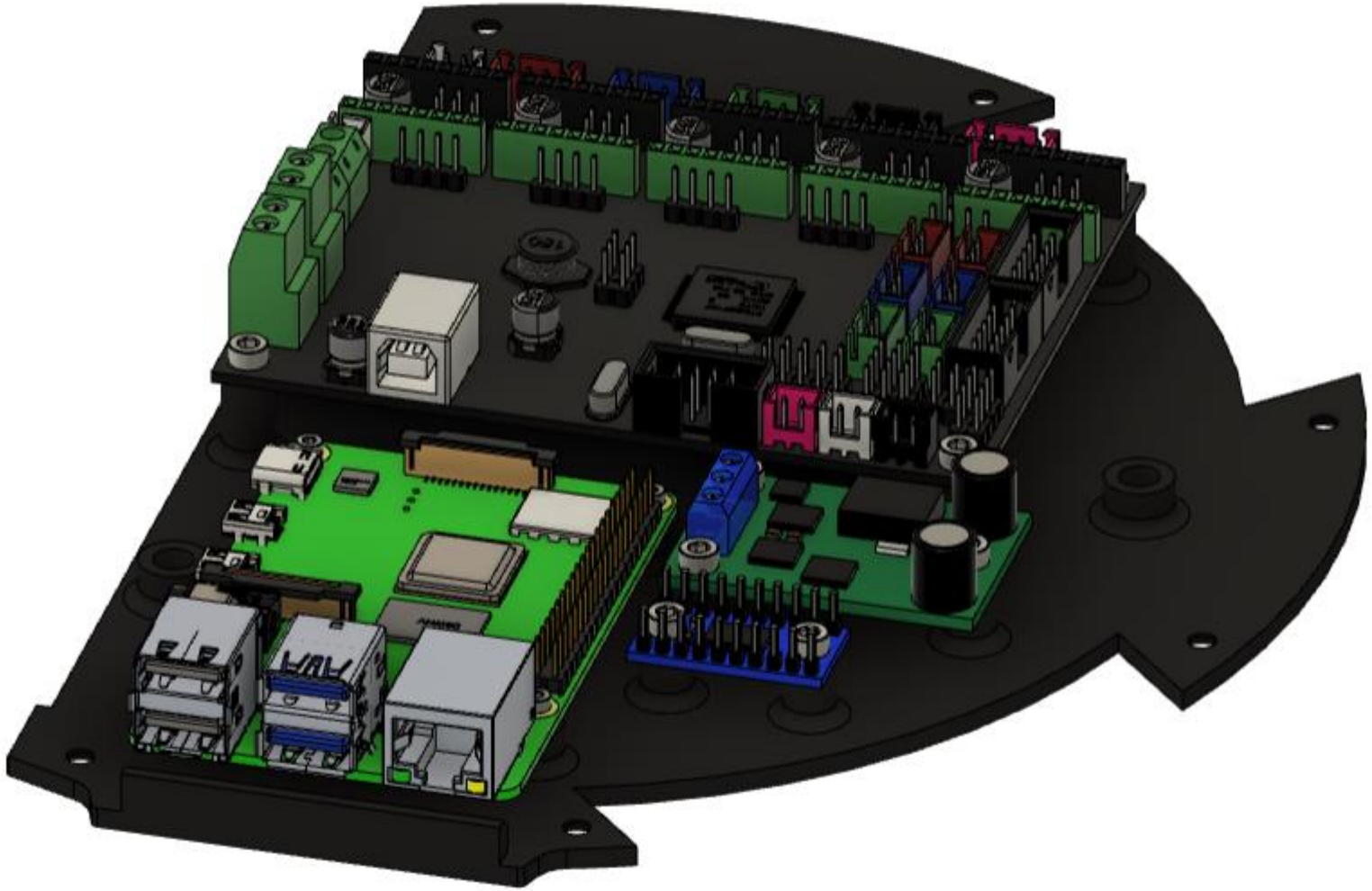
2. Connect the button and power socket to the power supply



4x M4x10
4x M4 nut

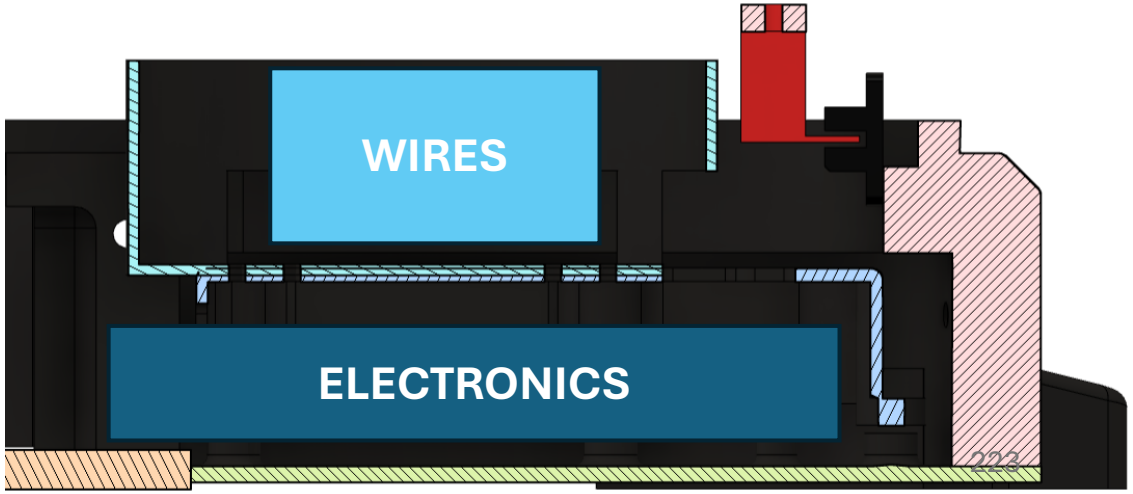


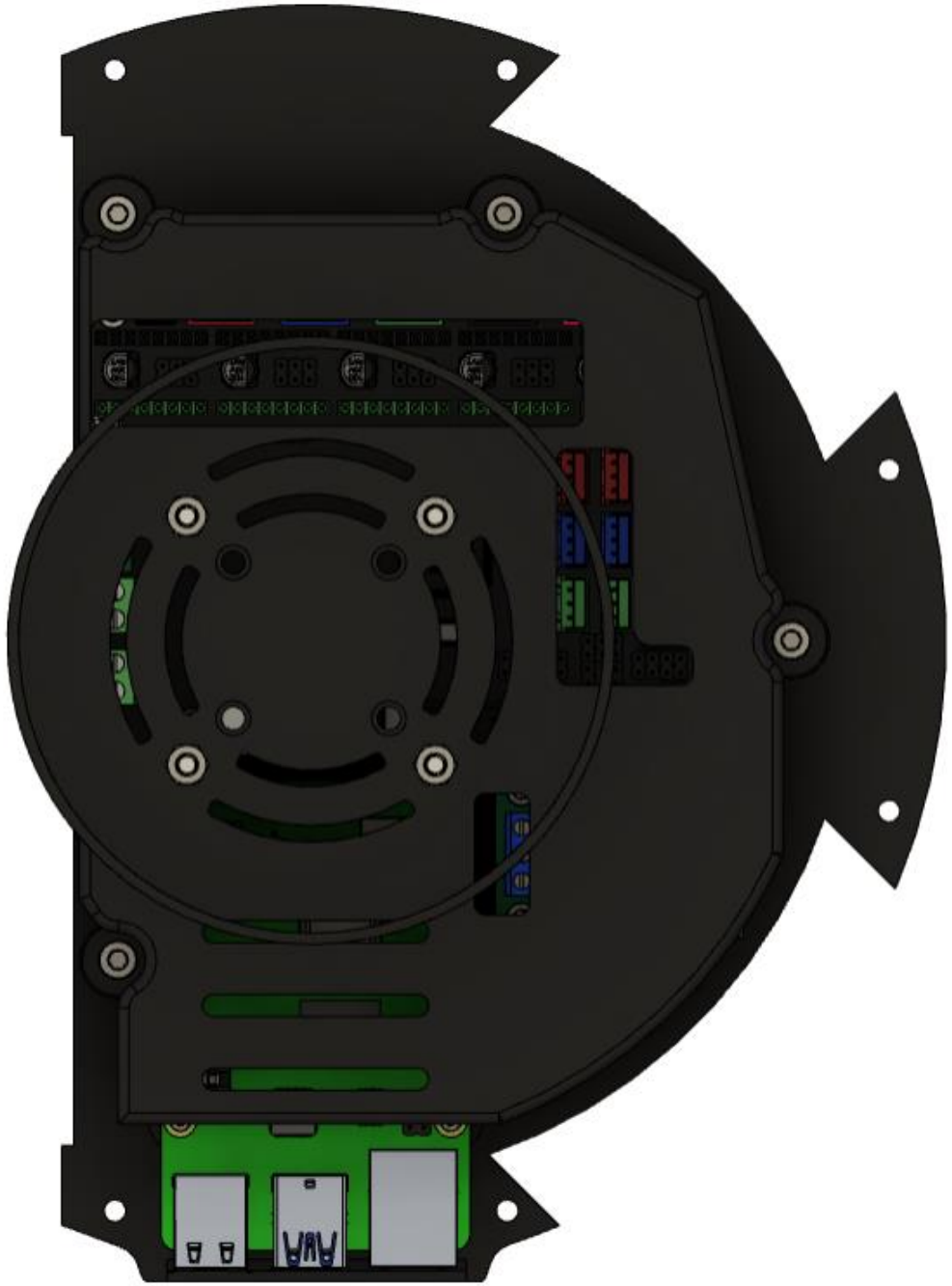
4x M2x6
9x M3x6



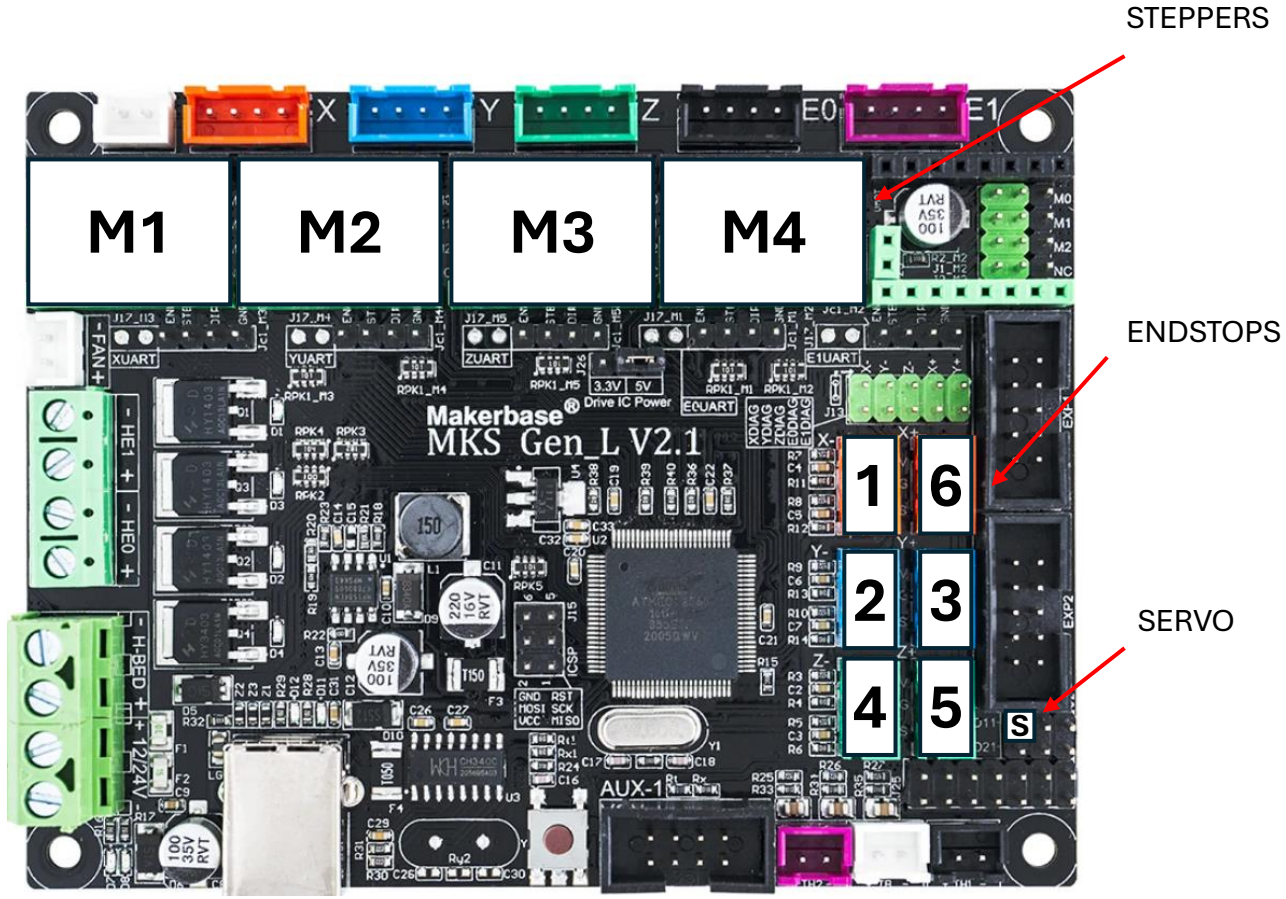
4x M4x10

These parts serve as a cover for the electronics and wires. It is important to ensure that the wires are being kept away from the endstop of joint 1 and its trigger to prevent possible entanglement.



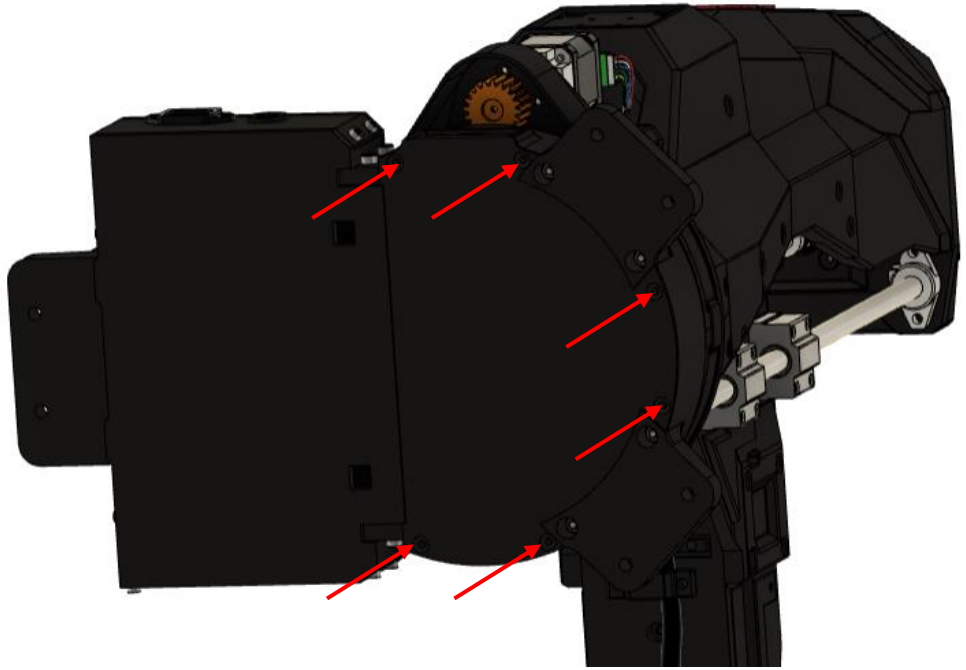


Connect the steppers, endstops and servo. The wires should go through the round enclosure mounted on top of the electronics cover.



6x M4x10

Place the robot on a table and rotate joint 1 by +90 or -90 degrees.



Note:
Joint 1 has to be rotated 90 degrees for the electronics and bottom cover to be attached to the robot (otherwise the sensor trigger of joint 1 will prevent the electronics to be inserted and mounted).

SOFTWARE

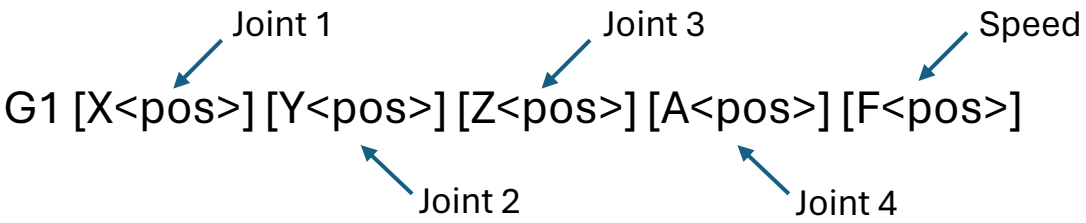
MARLIN

Since the robot uses a motherboard and software designed for 3d printers and CNC machines, it uses a similar control method - gcode.
G-code commands are instructions used to control the movement and operation of the robot. They dictate actions such as joints movements, positioning, dwell times, and other functions essential for task execution. You can find the complete list of commands here:
https://marlinfw.org/meta/gcode/
Note that the majority of the commands will not be used as they require other peripherals (e.g. temperature sensors) or are not necessary for this application.

G1 - Move
G4 - Dwell
G90 - Absolute Positioning
G91 - Relative Positioning
G92 - Set Position
M17 - Enable Steppers
M18 - Disable Steppers
M114 (R)- Get Current Position
M119 - Endstop States
M203 - Set Max Speed
M204 - Set Acceleration
M280 - Servo Position
M410 - Quickstop

G1 – MOVE COMMAND

This command is used for all the robot movements. This command defines the movement of each joint using degrees (for rotational joints) or millimeters (for linear joints).



Only the moving joints have to be defined, for example:
G1 X90 Y0 Z30 A0 F500 ✗
G1 X90 Z30 F500 ✓

If the speed is not defined, the movement will proceed with the last used speed (if no velocities have been defined since the robot has been turned on, the default speed will be used).

This gcode command results in a joint interpolated motion meaning all the joints start and stop at the same time. This means that the joint with the longest travel distance will be moving with the maximum defined speed while others will be moving slower.

NOTE: REMEMBER THAT THE G90 AND G91 COMMANDS AFFECT THE MOVE COMMAND!

G4 – DWELL
Stops the robot for a defined time (time given in milliseconds).

G4 <Time>

Example:
G4 P1000 - stops the robot for 1 second

G90 – ABSOLUTE POSITIONING

Changes the move type to absolute coordinates movements

G90

NOTE: A SMALL DELAY (G4) SHOULD BE INCLUDED BEFORE AND AFTER THIS COMMAND TO MAKE SURE THE POSITIONING WILL BE LOADED CORRECTLY

G91 – RELATIVE POSITIONING

Changes the move type to relative coordinates movements

G91

NOTE: A SMALL DELAY (G4) SHOULD BE INCLUDED BEFORE AND AFTER THIS COMMAND TO MAKE SURE THE POSITIONING WILL BE LOADED CORRECTLY

G92 – SET POSITION

Overwrites the current coordinates with user defined coordinates (used in homing scripts to set the home position).

G92 [X<pos>] [Y<pos>] [Z<pos>] [A<pos>]

NOTE: A SMALL DELAY (G4) SHOULD BE INCLUDED BEFORE AND AFTER THIS COMMAND TO MAKE SURE THE POSITIONS WILL BE LOADED CORRECTLY

M17 – ENABLE STEPPERS
Enables the steppers (if they have been disabled)

M17

M18 – DISABLE STEPPERS

Disables the stepper. Can be used only after the steppers have already been stopped (using M410). The stepper drivers have a built-in power saving function which disables the power after a short while. Use this command if you want to move the robot by hand after it moved using a gcode command.

M18

M114 (R) – GET POSITION
Returns the position of the robot joints (according to the stored values).

M114 - returns the position based on the current gcode movement line. **This means that it will not return any positions between two gcode move commands!**

M114 R – returns the current **CALCULATED** joint positions (based on the defined motion profile).

***NOTE: M114 R IS MUCH MORE USEFUL SINCE IT GIVES AN ESTIMATION OF JOINT POSITIONS IN REAL TIME*.**

M119 – ENDSTOP STATES
Returns the endstop states – “TRIGGERED” or “open”.
This function is used in homing scripts to stop the robot (joint 1 and joint 4) whenever a “TRIGGERED” state is detected.

Example:
Sent:
M119

Response:
x_min = open
y_min = open
y_max = TRIGGERED
z_min = TRIGGERED
z_max = open
a_min = open

M203 – SET MAX SPEED
Sets the maximum joint speed.

M203 [X<units/s>] [Y<units/s>] [Z<units/s>] [A<units/s>]

NOTE: IT IS ADVISED THAT THE MAX SPEED DOES NOT EXCEED 3500 (FOR THE CURRENT MICROSTEPPING VALUES). A MUCH HIGHER MAXIMUM VELOCITY IS POSSIBLE WITH LOWER (<32) MICROSTEPPING

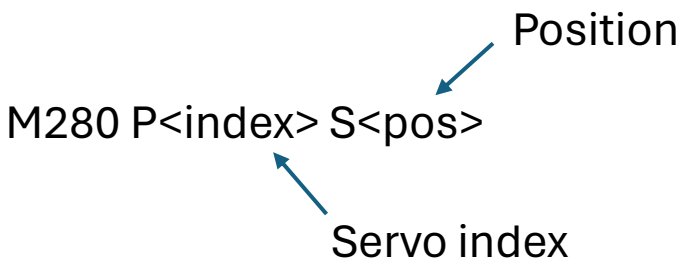
M204 – SET ACCELERATION
Sets the acceleration for all joints

M203 S<units/s>

NOTE: IT IS ADVISED THAT THE ACCELERATION DOES NOT EXCEED THE VALUE OF 35. HIGHER ACCELERATION VALUES MEAN BIGGER ARM VIBRATIONS.

M280 – SERVO POSITION

Moves the servo to an absolute position



Example:

M280 P0 S120 – moves servo attached to pin 1 to position 120.

NOTE: IN THE CURRENT APPLICATION THE SERVO IS ATTACHED TO PIN 1 SO P0 SHOULD BE USED.

M410 – QUICKSTOP
Stops all joints.

M410

NOTE: THIS COMMAND DOES NOT DISABLE THE STEPPERS NOR THE POWER SUPPLIED TO THEM. USE M18 TO DISABLE STEPPERS.

SERIAL COMMUNICATION

The MKS Gen L2.1 motherboard is communicating with the Raspberry Pi 4 using serial communication.
By default, the motherboard should be detected on the “ttyS0” port. The baudrate should be set to 250000. The timeout can be different than 0.2s but remember that it should be big enough to ensure that a whole command can be sent before the timeout happens. On the other hand, a too high value for the timeout would introduce unnecessary delays in the system.

A very simple script that sends gcode commands over serial communication can be seen below.

```python
import serial

#Open serial com
ser = serial.Serial('/dev/ttyS0', baudrate=250000, timeout=0.2)

try:
    while True:
        # Get user input
        command = input("Enter G-code command (e.g., G1 X10 F500): ")

        # Send command to the MKS board
        ser.write((command + '\n').encode())

        response = ser.read(1024).decode()
        print(response, end='')

except KeyboardInterrupt:
    ser.close()
    print("\nSerial port closed. Exiting...")
```

← Define serial communication

← Write user’s input to serial

← Receive and print response

FORWARD KINEMATICS

This script calculates the TCP (Tool Center Point) given specific joint values.

```
def scara_FK(theta_1, d_1, theta_3, theta_4):
```

There are two offsets included which correspond to the distance from the base of the robot to the starting point of joint 2 (123mm) and the vertical distance from start of joint 2 to the bottom of the gripper mount (-13.7mm).

```
d_1 = d_1 + 123 - 13.7 #BASE OFFSET=123mm, GRIPPER MOUNT OFFSET=-13.7mm
```

The frame transformation parameters are given by:

```
a_1_val, alpha_1_val, d_1_val, theta_1_val = 0, 0, d_1, -theta_1_rad+pi/2
a_2_val, alpha_2_val, d_2_val, theta_2_val = 329.498, 0, 0, 0
a_3_val, alpha_3_val, d_3_val, theta_3_val = 285.0, pi, 0, -theta_3_rad
a_4_val, alpha_4_val, d_4_val, theta_4_val = TCP_h, 0, TCP_v, theta_4_rad #TCP offsets
```

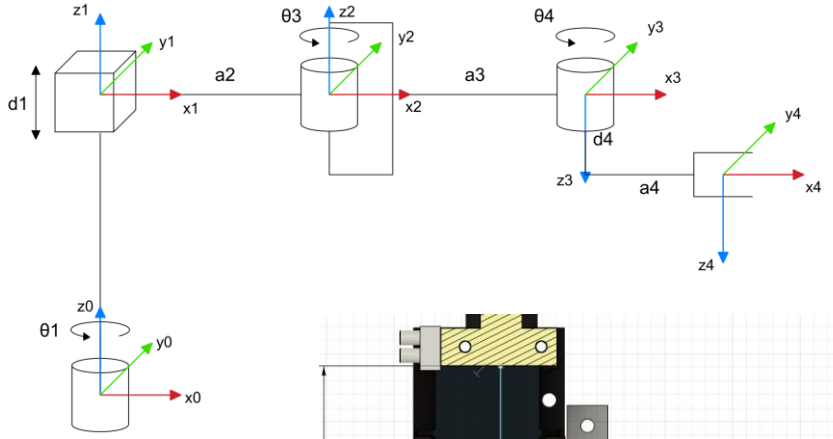
TCP offsets should be specified for the current end effector.

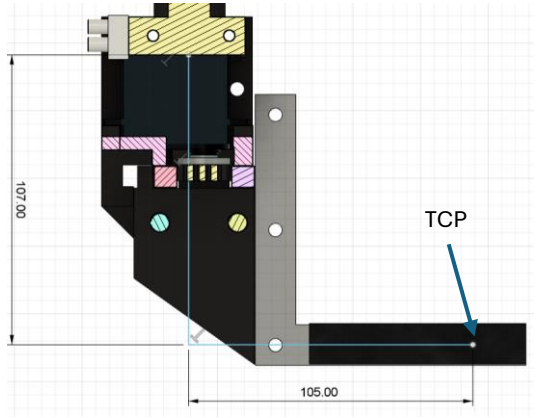
```
TCP_h = 105 #Horizontal offset
TCP_v = 107 #Vertical offset
```

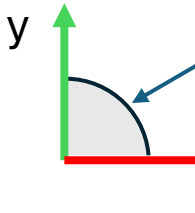
The function returns x,y,z coordinates and the end effector orientation (angle from the y axis)

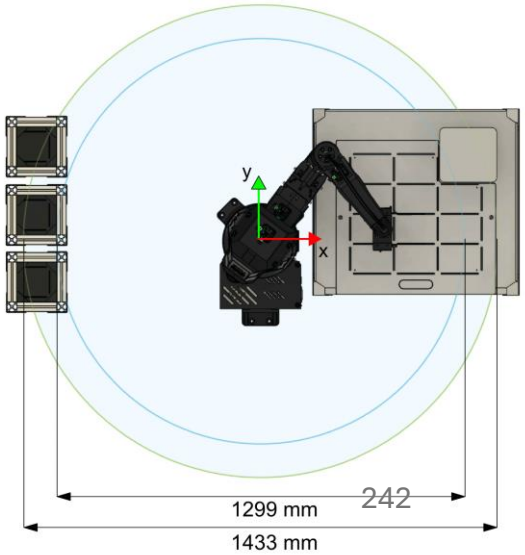
```
x = H_dh_04[0, 3]
y = H_dh_04[1, 3]
z = H_dh_04[2, 3]

g = theta_1 + theta_3 + theta_4

return x, y, z, g
```

If you want to get joint 4 orientation and position, use H_dh_03 to extract data in the same way.









INVERSE KINEMATICS

This script calculates the joint values given desired end effector x,y,z coordinates, end effector orientation and stored position (which includes both position and orientation

```
def calculate_scara_angles(x, y, z, gripper_or, pos):
```

position

gripper orientation

stored position with format: [0.0, 330.0, 00.0, 0.0] (example)

TCP offsets should be specified for the current end effector.

```
TCP_h = 105 #Horizontal offset
TCP_v = 107 #Vertical offset
```

The vertical positioning is only controlled with joint 2 which means that the z position is achieved by simply moving one joint (and accounting for the offsets).

```
z = z + l_4 - d_b + grip_off
```

The 2 link IK problem has two solutions, both of which are calculated in the script. The solutions are checked if the value of the calculated joint 3 angle does not exceed the joint's motion range.

Then, the cost (required travel) is calculated and the best solution is chosen. If both solutions have the same cost, the solution for a position with a smaller joint 1 value is chosen (it was assumed that the robot will be placed on the left side of the liquid handler)

The output of the function are joint values. If the function fails to output all 4 values, it means that the given coordinates are out of bounds (solution does not exist) for the given workspace.

```
return theta_1, d, theta_3, theta_4
```

NEAR MODE HOMING SCRIPT

This script should be used when the robot's stored position is incorrect (e.g. when the robot was moved by hand) or when the homing script which utilizes stored position has failed.

This script requires defined homing positions for each joint (joint starting positions). It is important that these values are accurate and reflect the real world starting positions. To find a joints starting position, comment out the homing sequence related to the other joints (leave only the sequence of the joint in question). Then the robot's position should be measured. It is advised to place the robot on a grid to allow for accurate measurements. Of course, multiple attempts can be made to ensure that the chosen starting position is correct.

```
#HOME POSITIONS
x_min_pos = -1.1
y_min_pos = 0.0
y_max_pos = 330.74
z_min_pos = -149.625
z_max_pos = 149.625
a_min_pos = 0.0
```

Note: The endstop notation is similar to the notation used in the gcode commands (refer to the G1 Move Command)
x_min – endstop of joint 1 -> **x_min_pos** – endstop 1
y_min – endstop of joint 2 -> **y_min_pos** – endstop 2
y_max – endstop of joint 2 -> **y_max_pos** – endstop 3
z_min – endstop of joint 3 -> **z_min_pos** – endstop 4
z_max – endstop of joint 3 -> **z_max_pos** – endstop 5
a_min – endstop of joint 4 -> **a_min_pos** – endstop 6

Homing requires different motion parameters (slower than normal moves). These include normal homing speed, precise homing speed and precise homing step. The homing algorithm will be explained on the next page.

```
#SPEED
normal_homing_speed = 500
x_precise_homing_speed = 300
x_precise_homing_step = 0.1
y_precise_homing_speed = 300
y_precise_homing_step = 0.5
z_precise_homing_speed = 300
z_precise_homing_step = 0.1
a_precise_homing_speed = 300
a_precise_homing_step = 0.2
```

The endstops have to be specified according to the chosen homing sequence.

```
#DEFINE HOME POSITIONS BASED ON DESIGNED HOMING SEQUENCE
home_positions = [x_min_pos, y_max_pos, z_min_pos, a_min_pos]

# List of specified endstops
specified_endstops = ['x_min', 'y_max', 'z_min', 'a_min']
```

NEAR MODE HOMING SCRIPT – P2

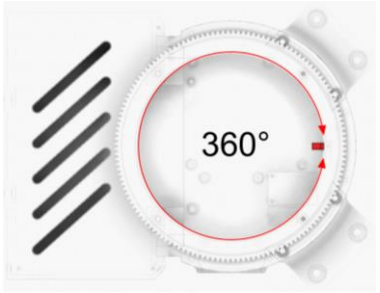
The sensor_check function returns the state for the given endstop (it uses the M119 command and finds the state for the current given endstop)

```
def sensor_check(ser, endstop):
```

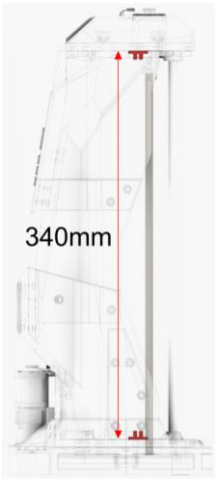
The homing_check function continuously checks the endstop state and stops the joint when the desired endstop gets triggered.

```
def homing_check(ser, endstop):
```

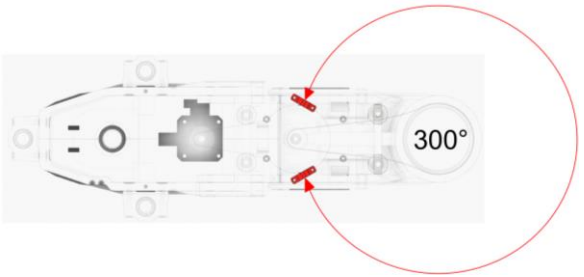
Joint 1



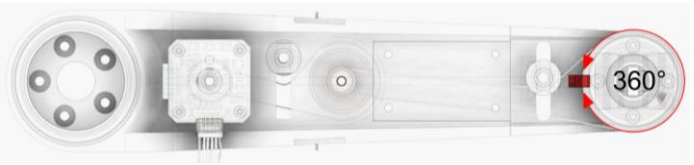
Joint 2



Joint 3



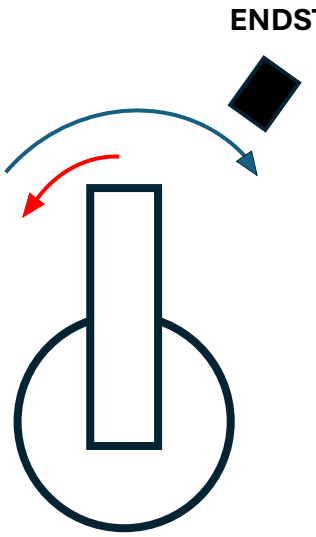
Joint 4



It is important to understand the difference between joint 1, joint 4 and joint 2, joint 3.
Joint 2 and joint 3 utilize built-in Marlin functions that stop the motor whenever their sensors gets triggered (only one direction is blocked).
This means that homing can be achieved through simply moving the joint in one direction until the endstop is encountered.
Joint 1 and 4 have only one endstop which creates a problem – if there is no stored position, it does not know on which side of the endstop it is currently.
Rotating in one direction until an endstop is triggered is not feasible due to cable limitations.

NEAR MODE HOMING SCRIPT – P3

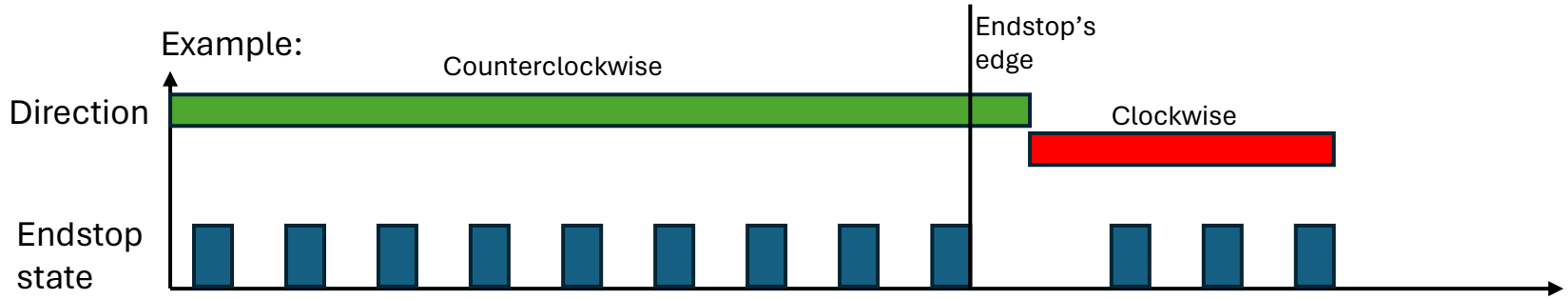
The idea behind the near mode homing is to place the robot close to the homing position by hand (backdrivability is not possible for joint 2 since it was designed to not be backdrivable). Then the script moves the joint back and forth (for joint 1 and 4) or continuously in a specified direction (for joint 2 and joint 3) in order to trigger the endstop. For example, the joint rotates 10 degrees counterclockwise and if the endstop is not triggered, it moves clockwise 20 degrees (there is a 5s timeout).



After this, the joint is roughly in the home position. Visually the robot seems to have perfectly homed the joint but keep in mind that very small deviations can lead to accuracy problems.
This is why small movements in the form of steps have been implemented to ensure precise homing.

```
def precise_homing(ser,axis,endstop,precise_homing_step,precise_homing_speed):
```

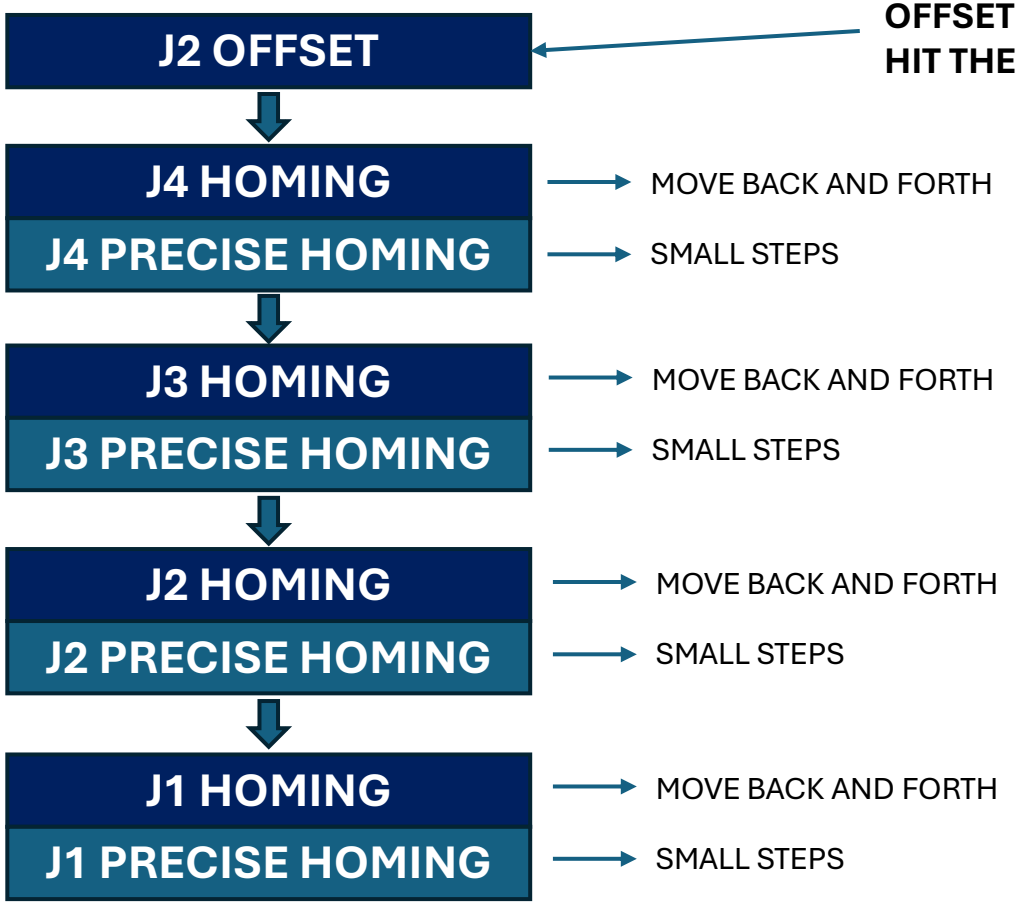
After the endstop is triggered, the joint starts moving in a specified direction until the state is “open” again (the endstop is not triggered anymore). This means that the joint has arrived at the very edge of the sensor with high precision. Then, all that is left to do is for the joint to do a few steps backwards (in the other direction from the original one) and the homing of the current joint is finished.



NOTE: **IF NEAR MODE HOMING FAILS THEN IT SHOULD SIMPLY BE REPEATED. YOU CAN ALSO CONSIDER TO CHANGE THE MOTION PARAMETERS (STEP SIZE AND SPEED). REMEMBER THAT THESE PARAMETERS IMPACT THE TIMEOUTS ETC.**

NEAR MODE HOMING SCRIPT – P4

The homing script follows a specific sequence defined by the user (depends on the application).



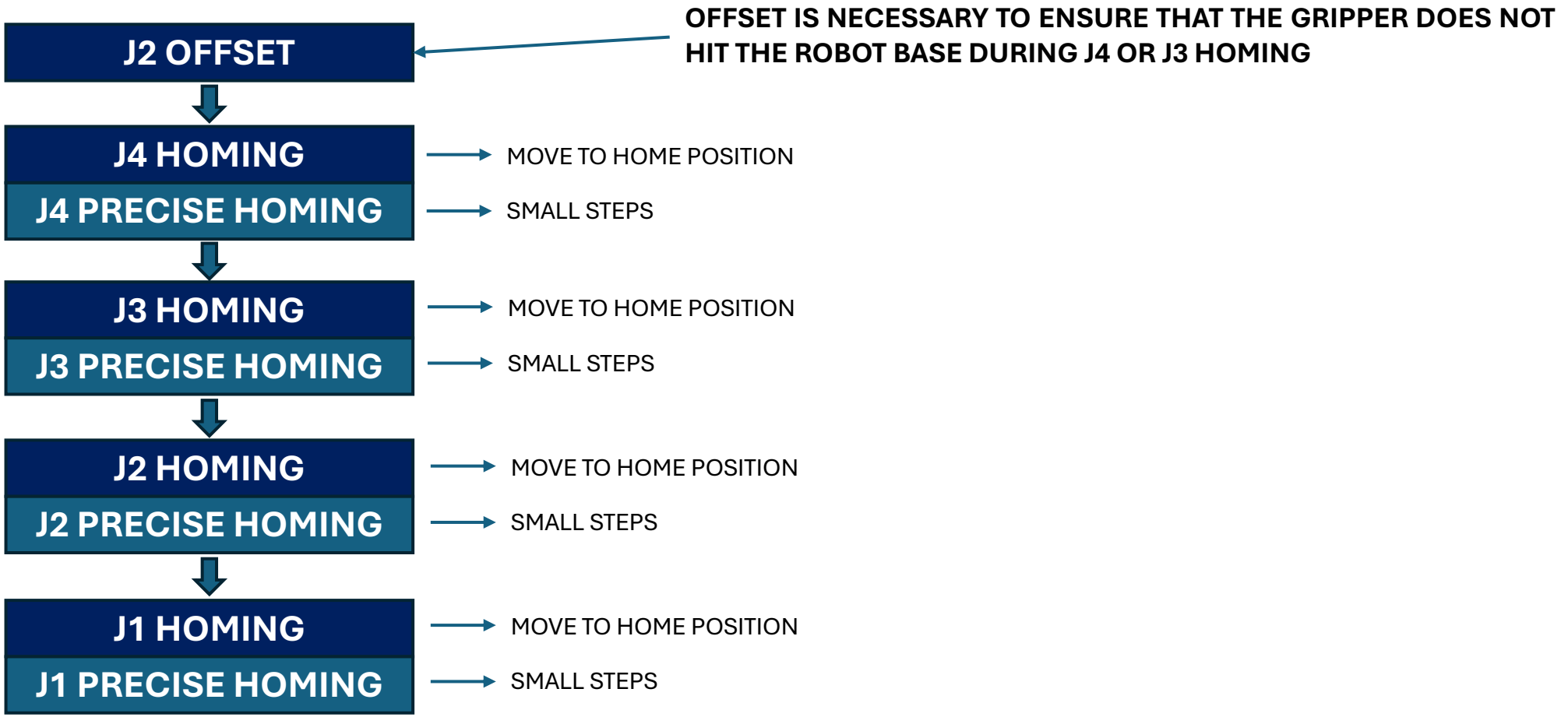
At the end of the sequence, the current joint values are overwritten with the home positions.
Additionally, as a safety measure, the status is checked for all specified endstops (M119 command). If an endstop state is not “TRIGGERED” then the robot has to be homed by the user again.

At the end of the script, the homing positions are written into a txt file that is used for storing the positions.

HOMING SCRIPT

The homing script is different from the near mode homing because it uses last stored positions as a reference. By knowing these positions, the robot does not have to guess on which side of the endstop the endstop trigger is located (important for joint 1 and 4).

This script utilizes the same sequence and logic but rather than moving the joint back and forth, it moves to the specified home position in order to trigger the endstop.



MOTORS STOP

To stop the motors the script has to interrupt the current open serial communication. This is achieved through simply closing the serial communication port and then starting a new one.

```
ser = serial.Serial('/dev/ttyS0', baudrate=250000, timeout=0.2)
ser.close()
# Start serial
ser = serial.Serial('/dev/ttyS0', baudrate=250000, timeout=0.2)
```

Then, a quickstop command is sent to the motherboard. This command stops the motor but does not disable the power.

```
ser.write('M410\n'.encode())
```

To disable the power, a new command has to be sent.

```
ser.write('M18\n'.encode())
```

After these two commands, the robot is stopped and the motors are disabled.