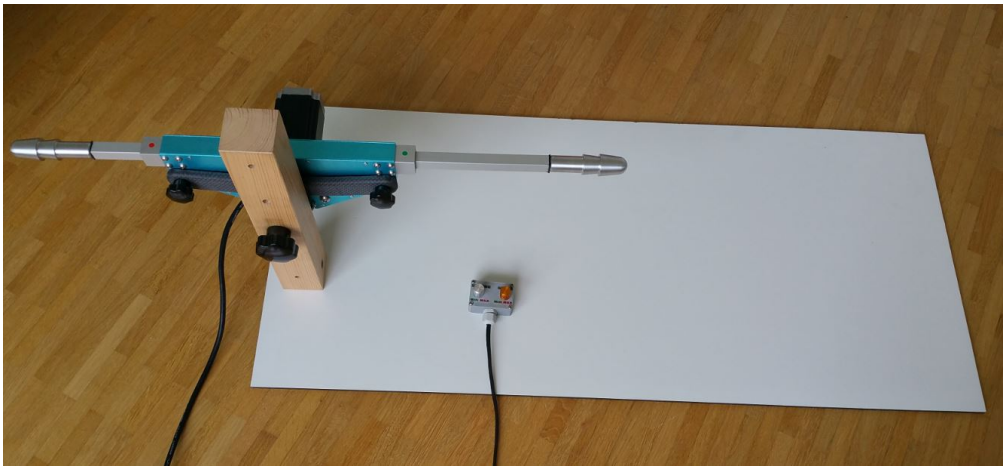
lm42p-diy

Release 1

Apr 21, 2023

Contents:



On this page you will find all the information you need to build the LoveMachine42People. Five parts are dedicated to detail all the operations of realization. You will find respectively the plans for the Machine, the Electrical Part, the Fixing Components, the Accessories and the Wooden Storage Box.

This page is under construction, so it may not contain all the details yet. But come back from time to time to see the progress.

This page takes me a lot of time to write, it would be a great support if you could make a small donation for its realization by clicking on the PayPal logo below or by subscribing to my youtube channel. That way, I will be all the more motivated to finish it quickly and it will allow me to know your interest in acquiring the plans of this machine.

Thank You a thousand times for your support!

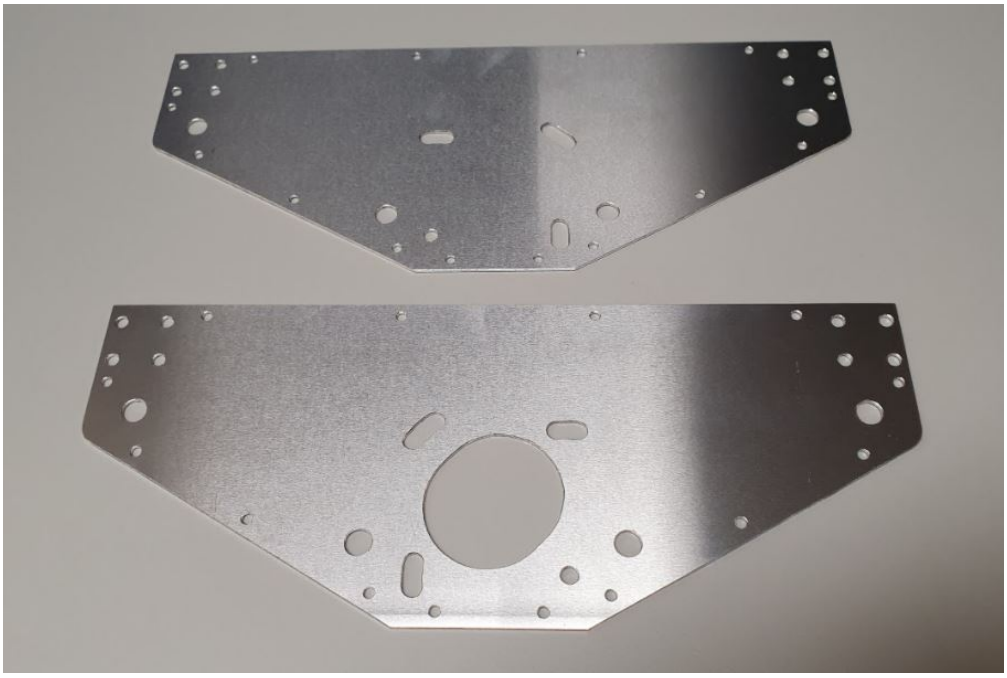
The puplication of this document is only dedicated to the manufacture of your personal machine. You can read it for educational purposes. Any use of this document for commercial purposes is strictly prohibited. The copy of all parts referring to this product (photos, videos, plans) is prohibited and must be requested to the author LM42P.

CHAPTER 1

Machine

1.1 Static Parts

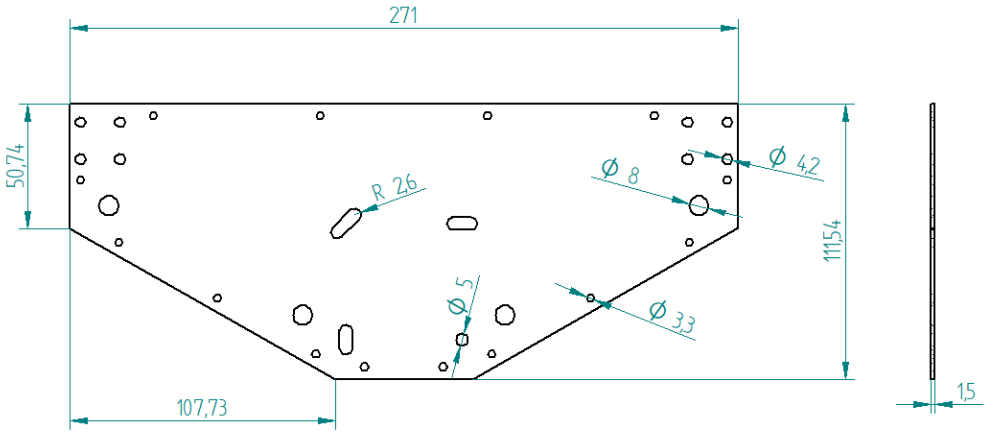
1.1.1 Casing



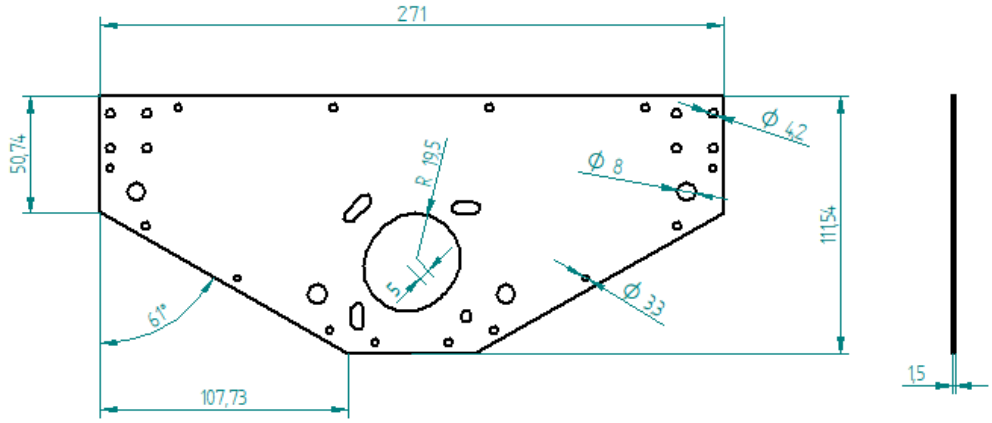
Video

This video shows how to make the Casing and the Casing Motor Side

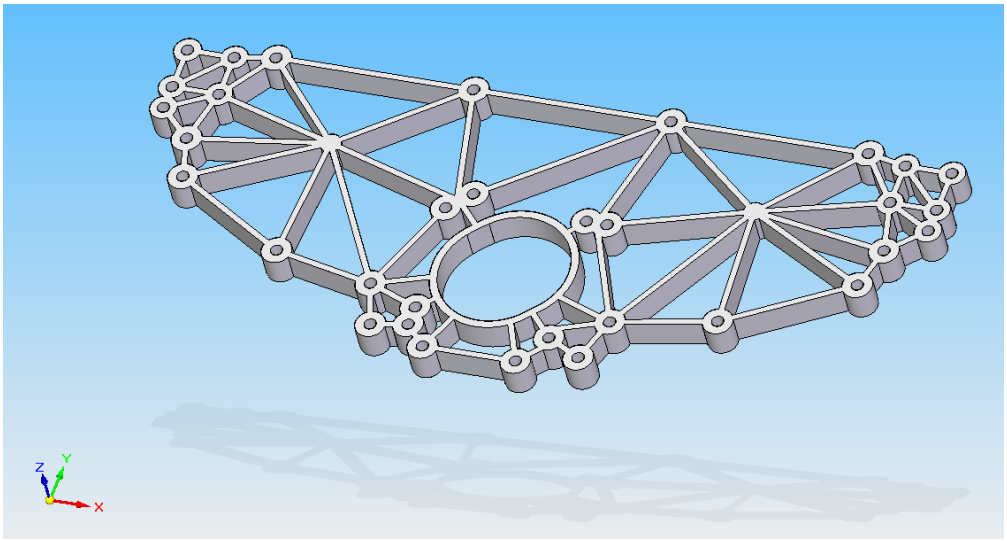
Casing



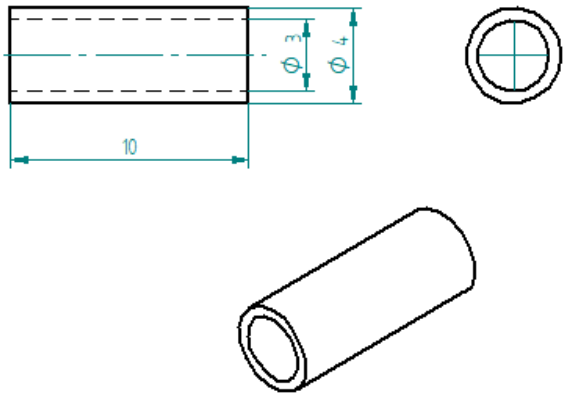
Casing Motor Side



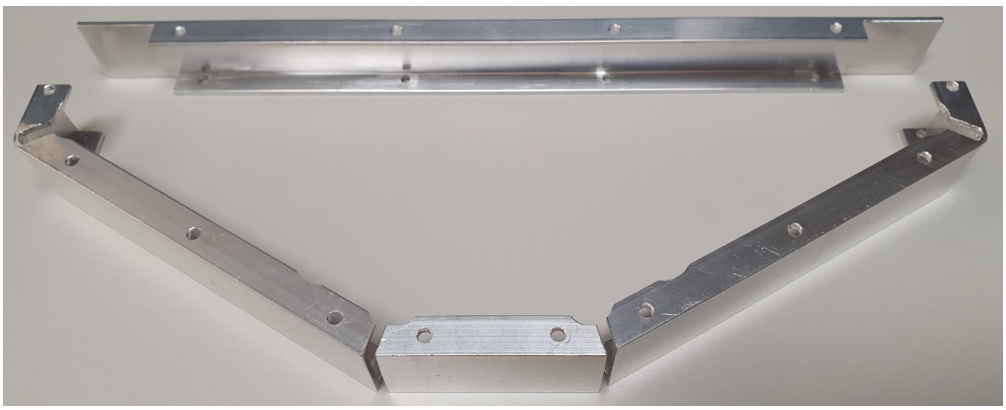
Jig Casing

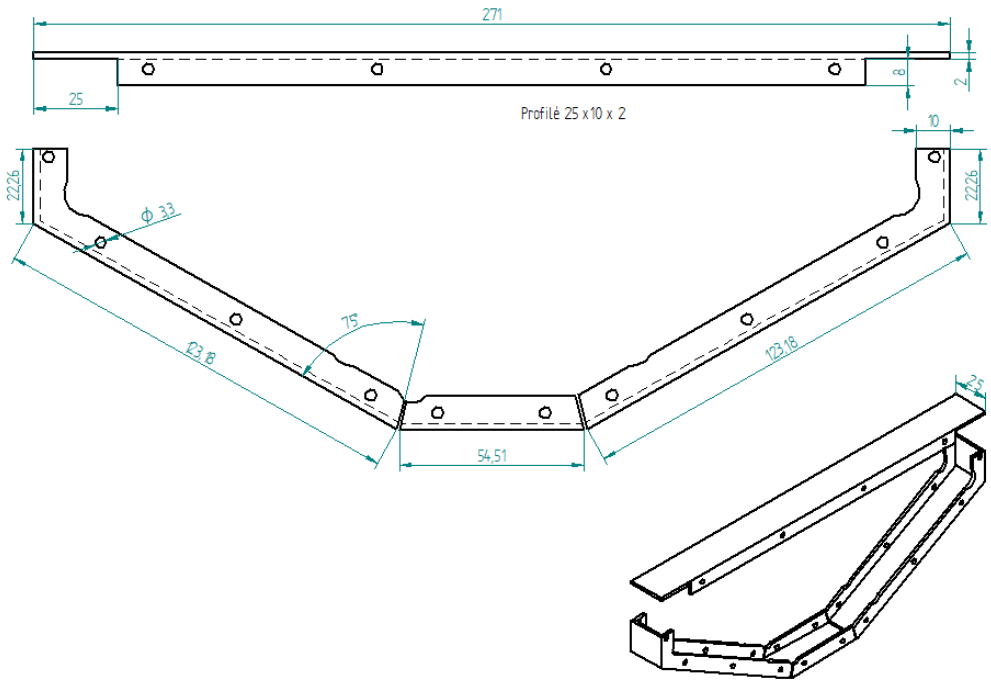


- STL file ->
- Drill Bush (to be hunted in jig) material : brass



1.1.2 Frame

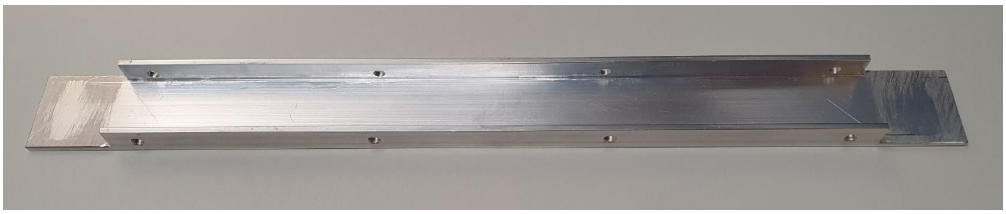




Video

This video shows how to make the Frame :

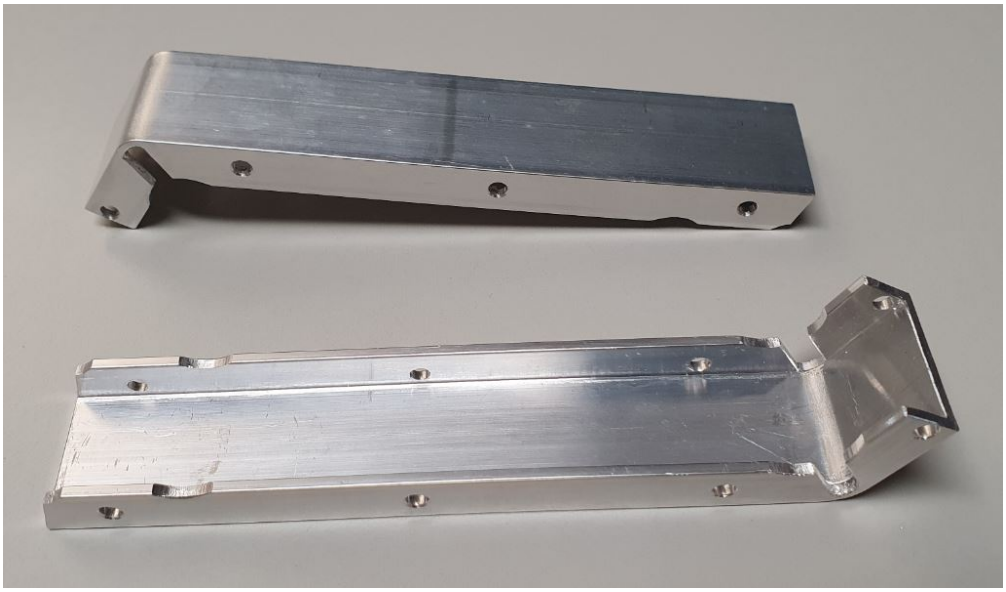
Large Frame



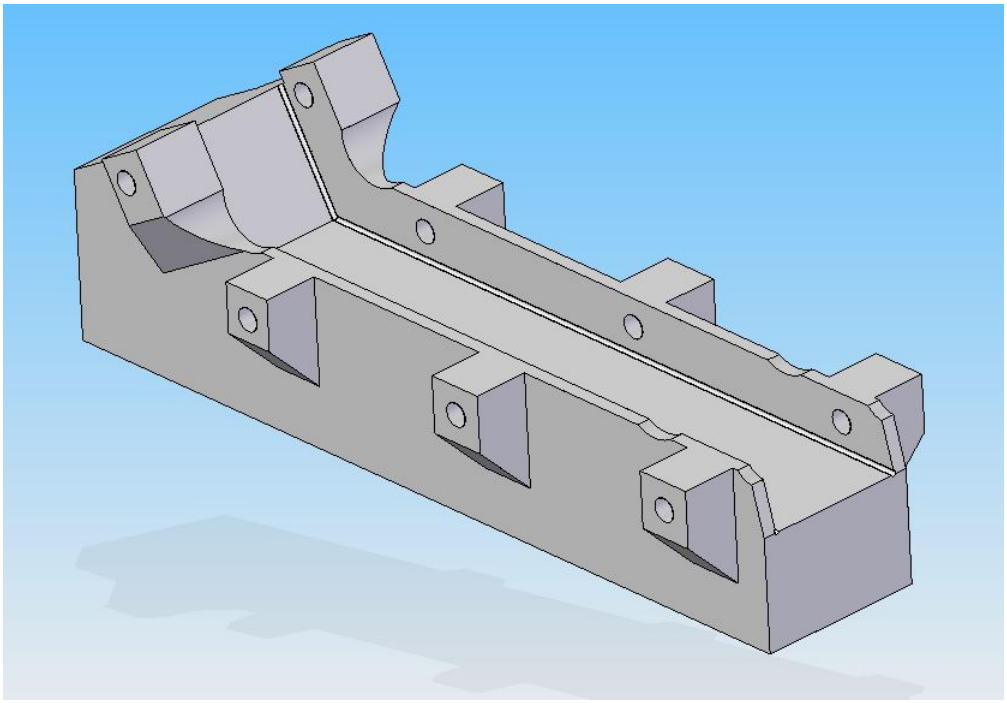
Jig Drill Large Frame

STL file ->

Medium Frame

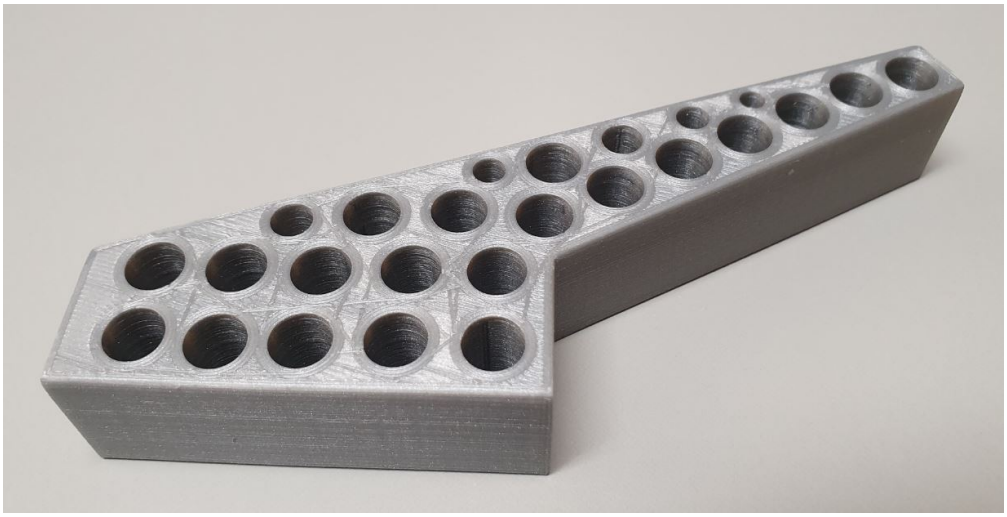


Jig Drill Medium Frame



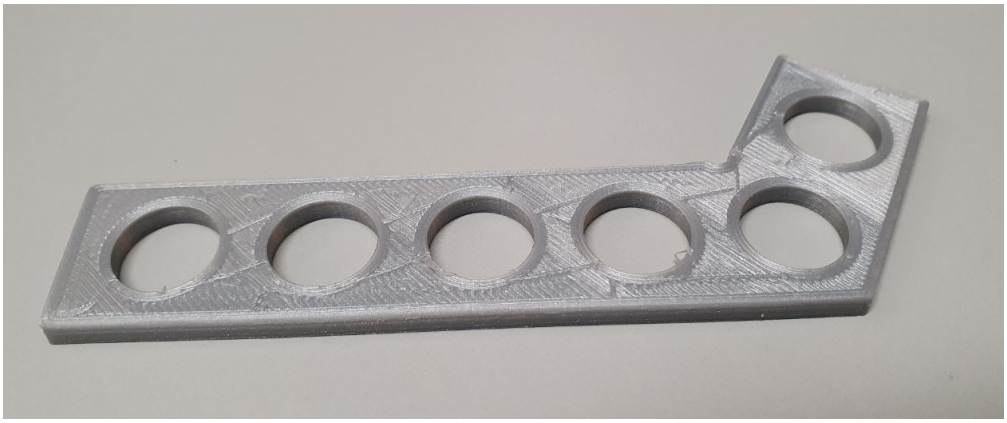
STL file ->

Jig Bend Medium Frame



STL file ->

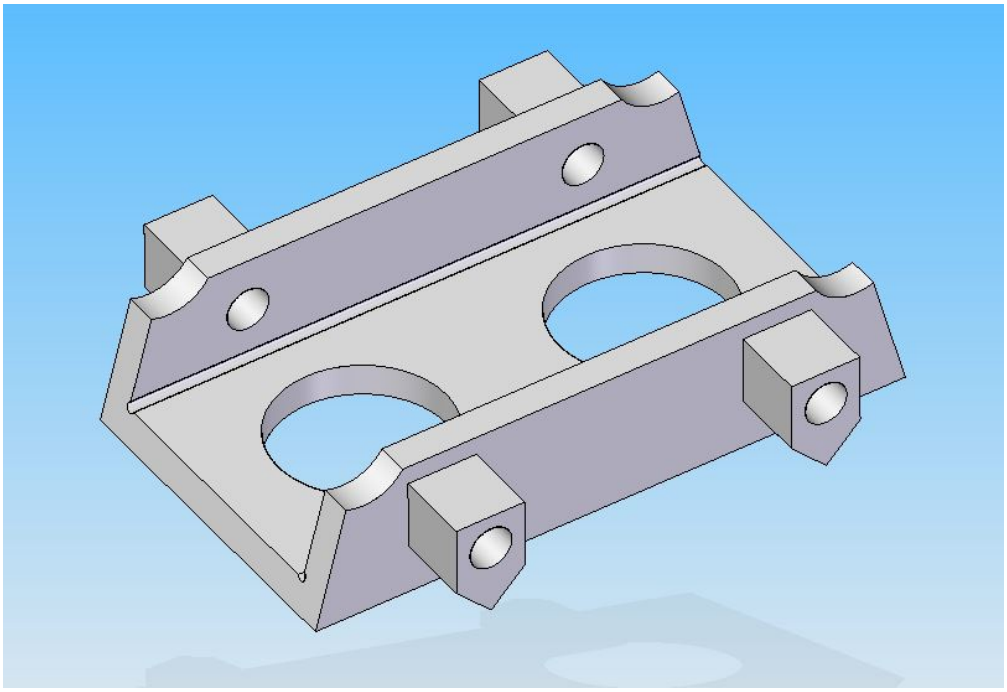
Angle Gauge Medium Frame



STL file ->

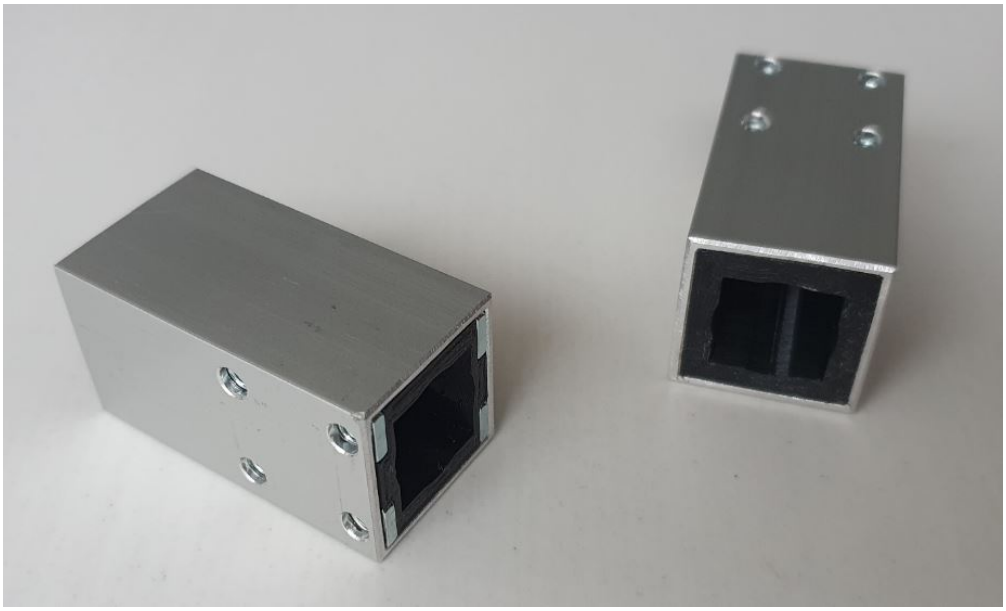
Small Frame

Jig Drill Small Frame



STL file ->

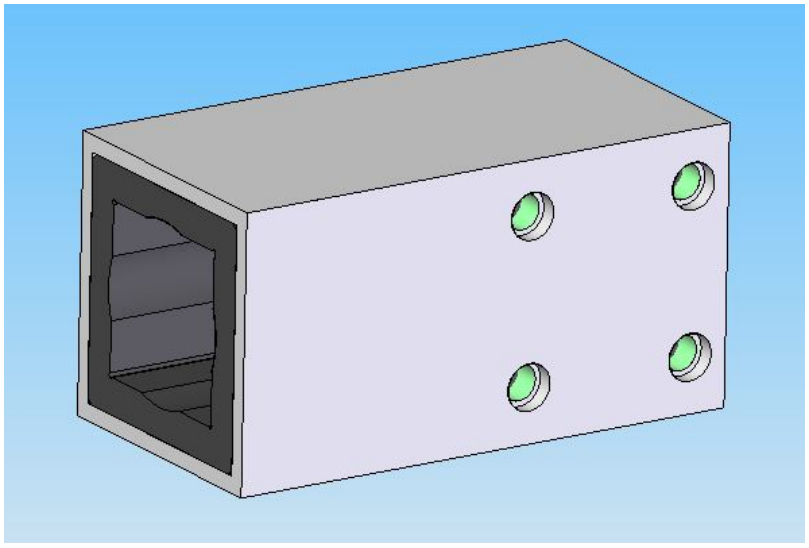
1.1.3 Bushing

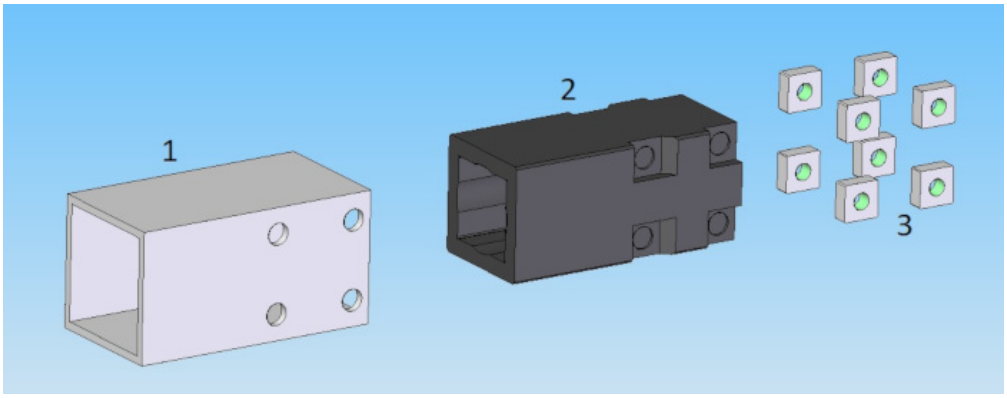


Video

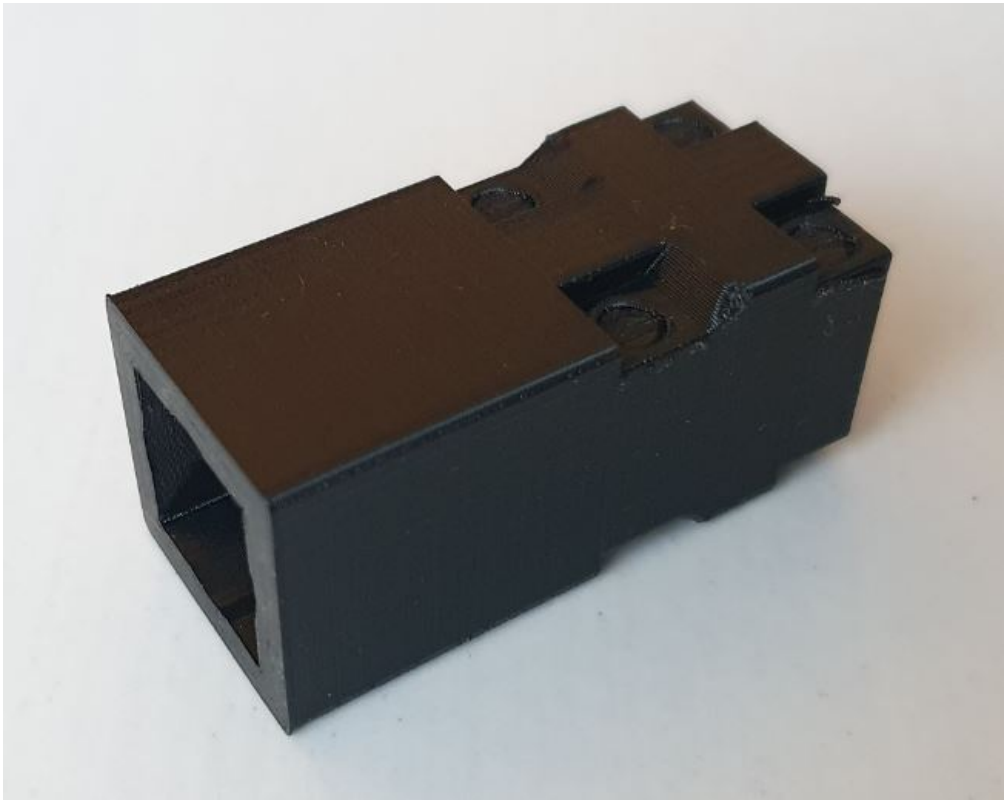
This video shows how to make the Bushing.

Quantity : 2x



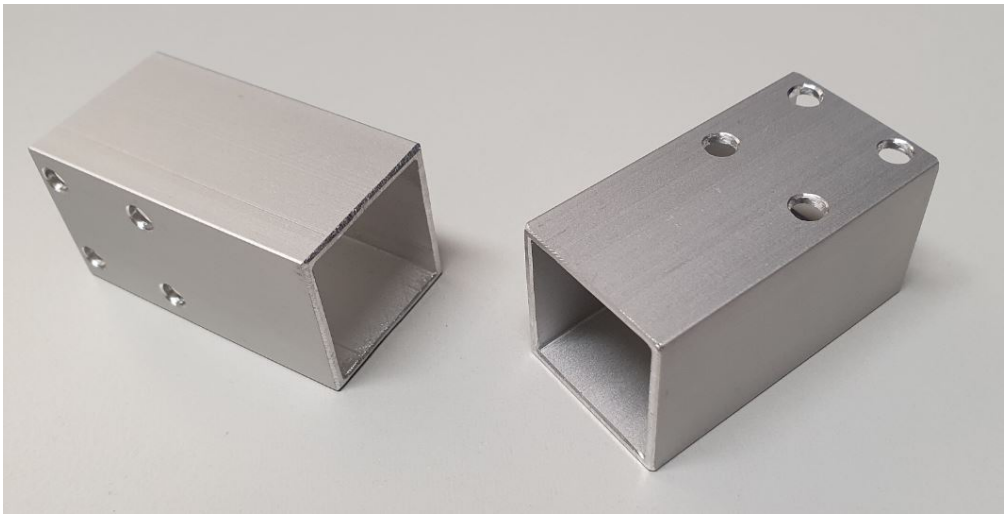


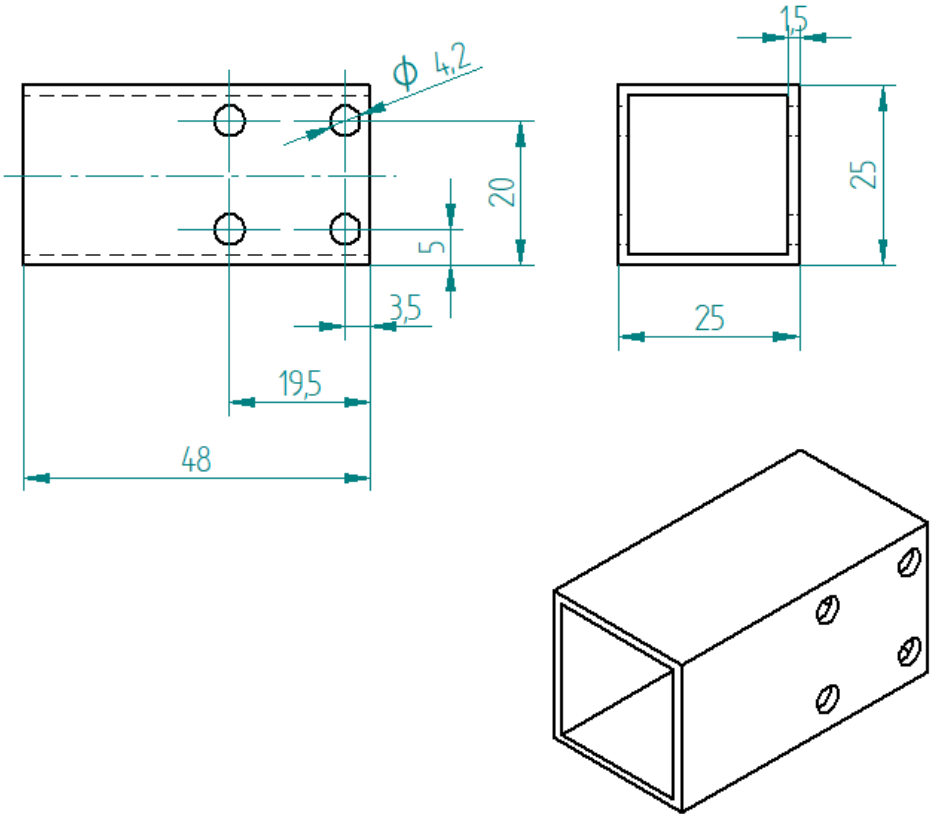
Bushing Sleeve



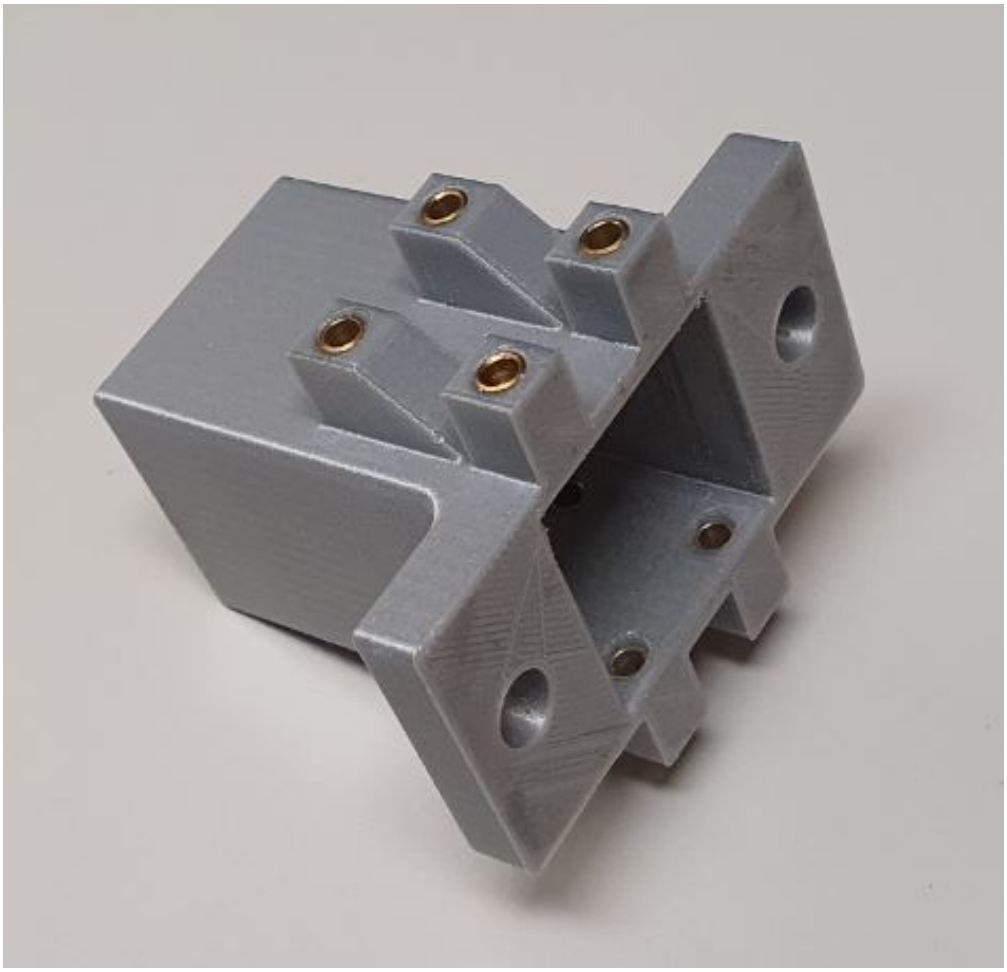
- Material : nylon
- STL file ->

Bushing Shell





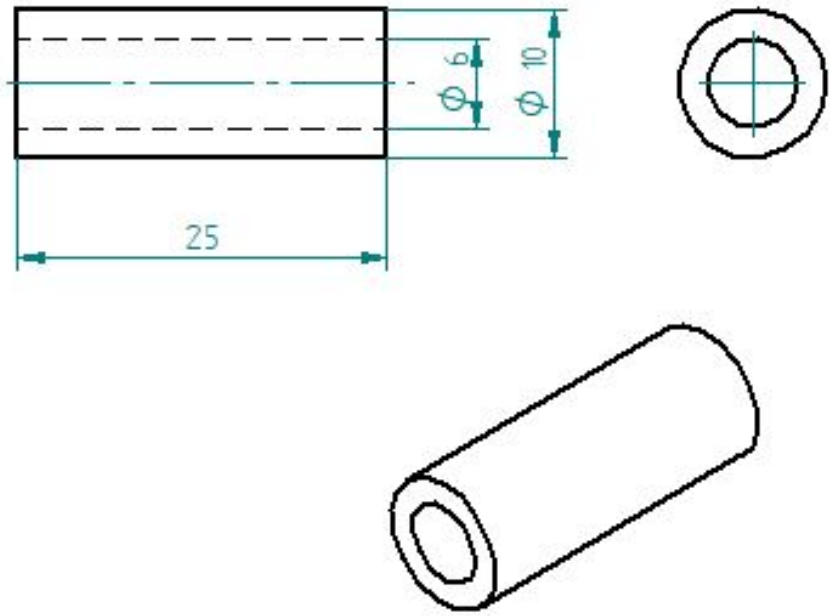
Jig Drill Bushing Shell



STL file ->

1.1.4 Spacer

This video shows how to make the Spacer.



1.2 Moving Parts

1.2.1 Listing Parts

Motor

Quantity : 1x

Timing Pulley

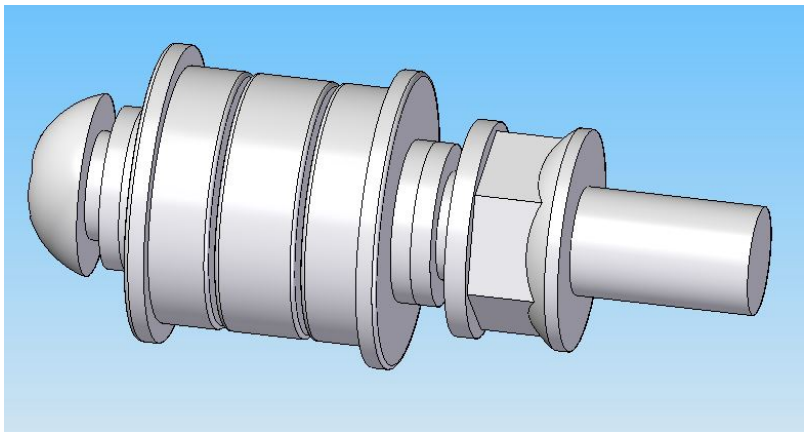
- Quantity : 1x
- Type : HTD3M Type 40T
- Size : hole diameter = 8mm width = 16mm
- Where to buy :

Timing Belt

Quantity : 1x

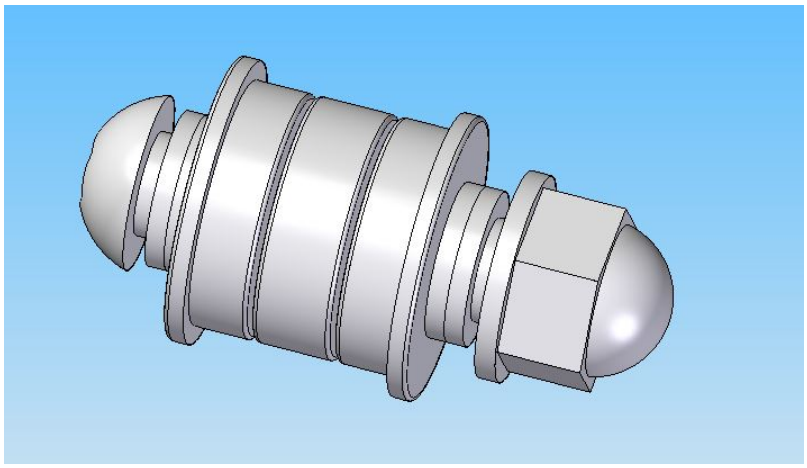
Idler Pulley Long Screw

Quantity : 2x



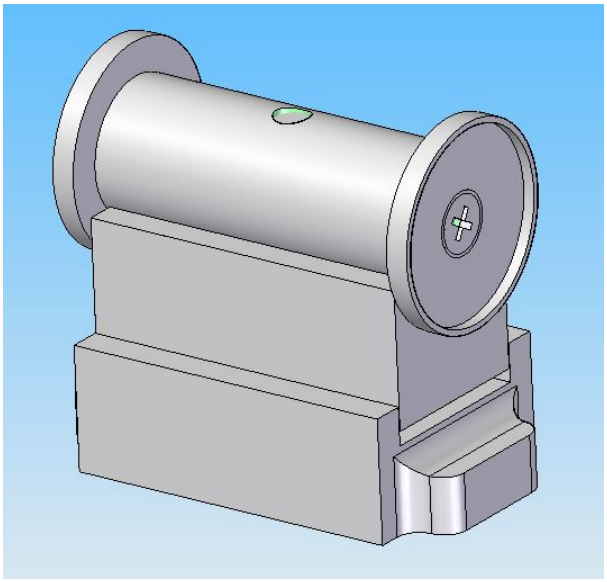
Idler Pulley Short Screw

Quantity : 2x



Belt Joint

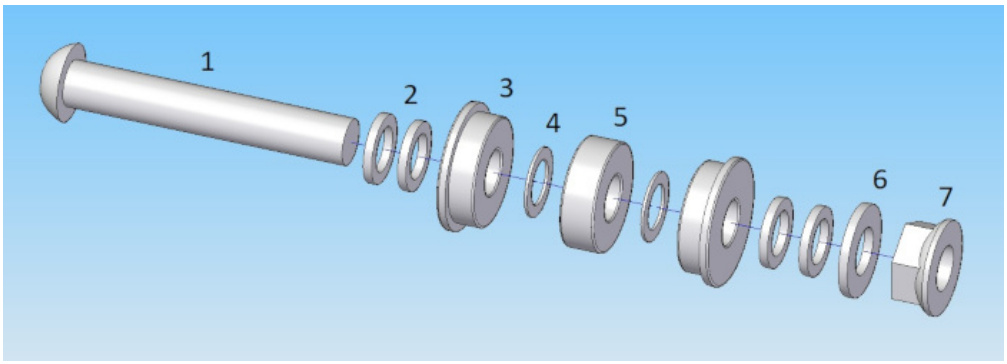
Quantity : 1x



Rod

Quantity : 1x or 2x

1.2.2 Idler Pulley Long Screw

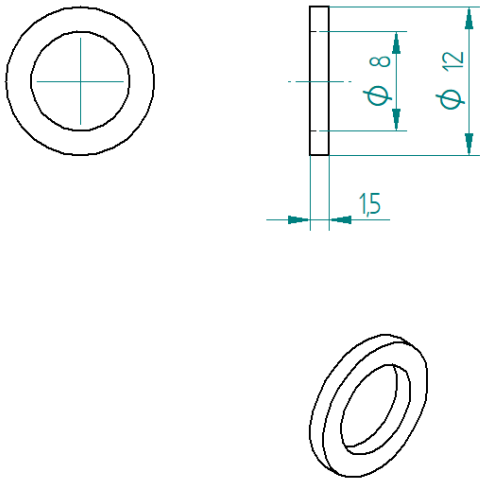


Listing Parts

1) Long Screw M8
    - Quantity : 1x
    - Length : 50mm
    - Material : stainless

2) Washer 8 x 12 x 1.5

- Quantity : 4x
- Material : aluminium



Note: Maybe better to use 1x Stainless steel Flat Washer Thickness 1mm (8x14x1mm) and 1x Stainless steel Flat Washer 0.5mm (8x14x0.5). 1.5mm doesn't exist. Because aluminum tends to crush when it's squeezed.

3) Flange Ball Bearing

- Quantity : 2x
- Type : F698ZZ
- Size : 8 x 19 x 6 mm

4) Flat Washer

- Quantity : 2x
- Thickness : 0.5mm
- Size : 8 x 12 x 0.5 mm
- Material : stainless

5) Bearing

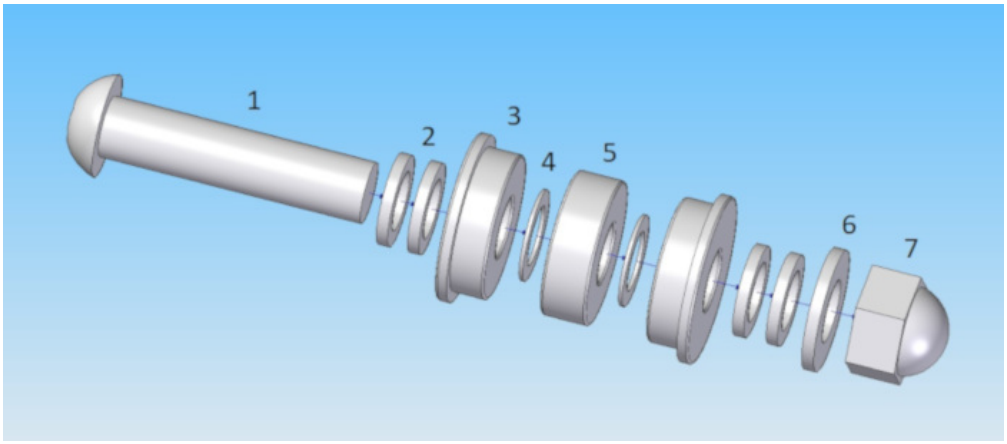
- Quantity : 1x
- Type : 698ZZ
- Size : 8 x 19 x 6 mm

6) Washer M6

- Quantity : 1x
- Material : stainless
- Size : 8 x 16 x 1.6 mm

7) Nuts M8 Stop
    - Quantity : 1x
    - Material : stainless

1.2.3 Idler Pulley Short Screw

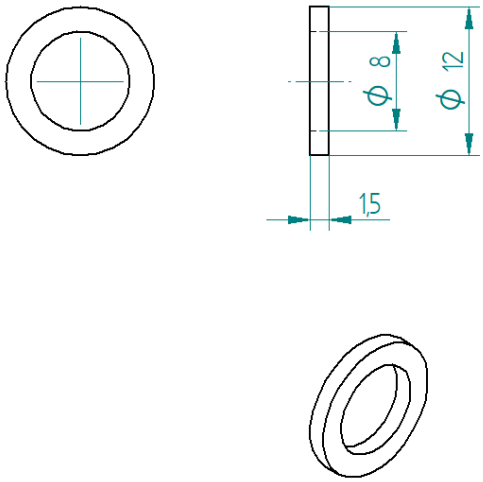


Listing Parts

1) Short Screw M8
    - Quantity : 1x
    - Length : 35mm
    - Material : stainless

2) Washer 8 x 12 x 1.5

- Quantity : 4x
- Material : aluminium



Note: Maybe better to use 1x Stainless steel Flat Washer Thickness 1mm (8x14x1mm) and 1x Stainless steel Flat Washer 0.5mm (8x14x0.5). Because aluminum tends to crush when it's squeezed.

3) Flange Ball Bearing

- Quantity : 2x
- Type : F698ZZ
- Size : 8 x 19 x 6 mm

4) Flat Washer
    - Quantity : 2x
    - Thickness : 0.5mm
    - Size : 8 x 12 x 0.5 mm
    - Material : stainless

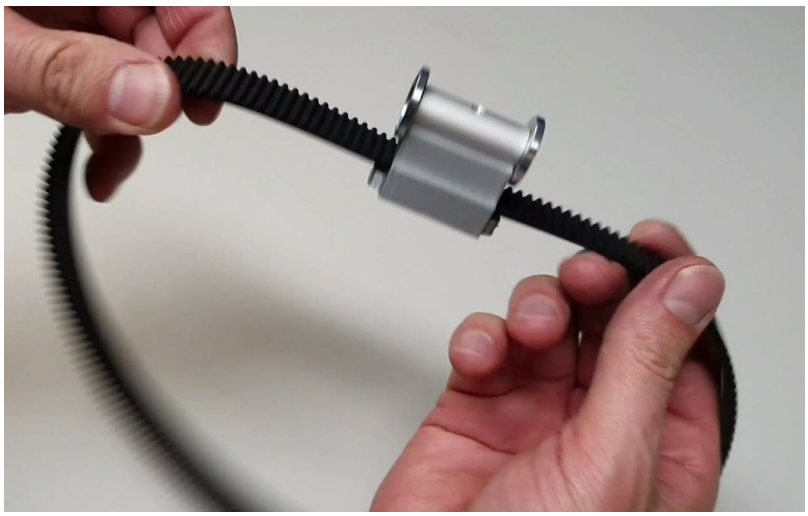
5) Bearing
    - Quantity : 1x
    - Type : 698ZZ
    - Size : 8 x 19 x 6 mm

6) Washer M8
    - Quantity : 1x
    - Material : stainless
    - Size : 8 x 16 x 1.6 mm

7) Cap Nuts M8

- Quantity : 1x
- Material : stainless

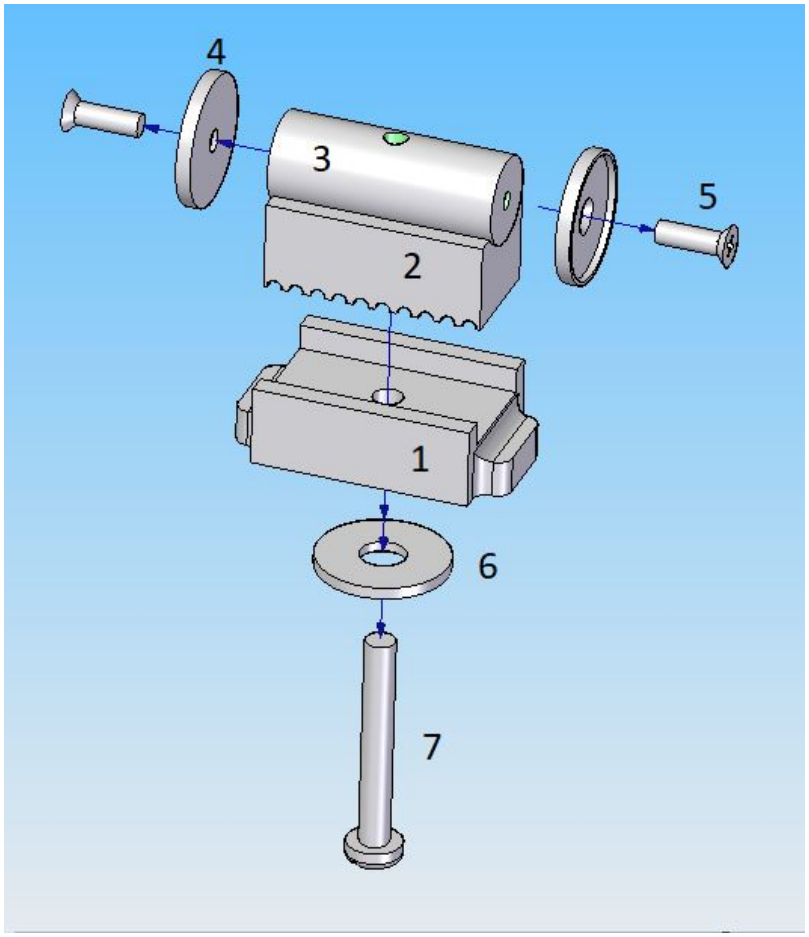
1.2.4 Belt Joint



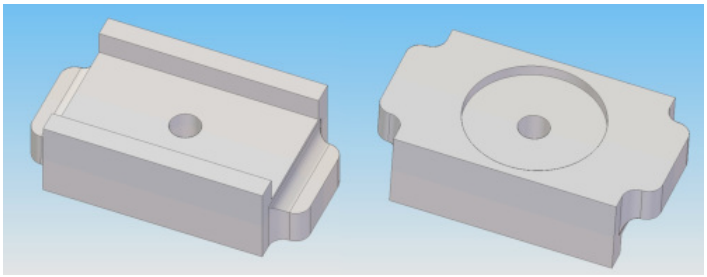
Video

This video shows how to make the Belt Joint

Listing Parts

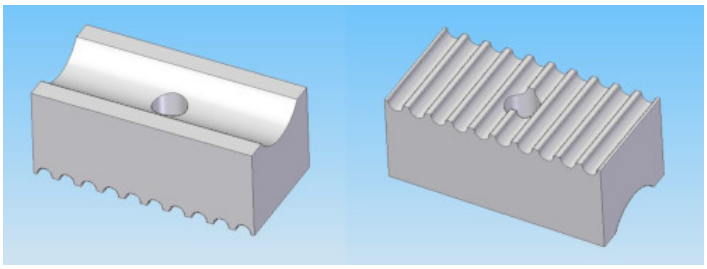


1. Base Belt Joint



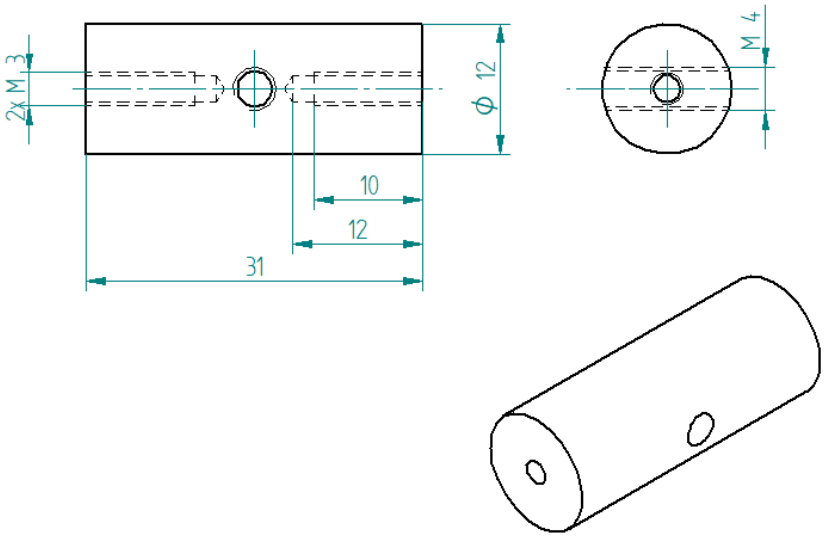
- Quantity : 1x
- Material : 3D printed PLA (innerfill = 100%)
- STL file ->

2. Counterpart Base Belt Joint



- Quantity : 1x
- Material : 3D printed PLA (innerfill = 100%)
- STL file ->

3. Spacer Metal Disc
   - Quantity : 1x
   - Material : Aluminium



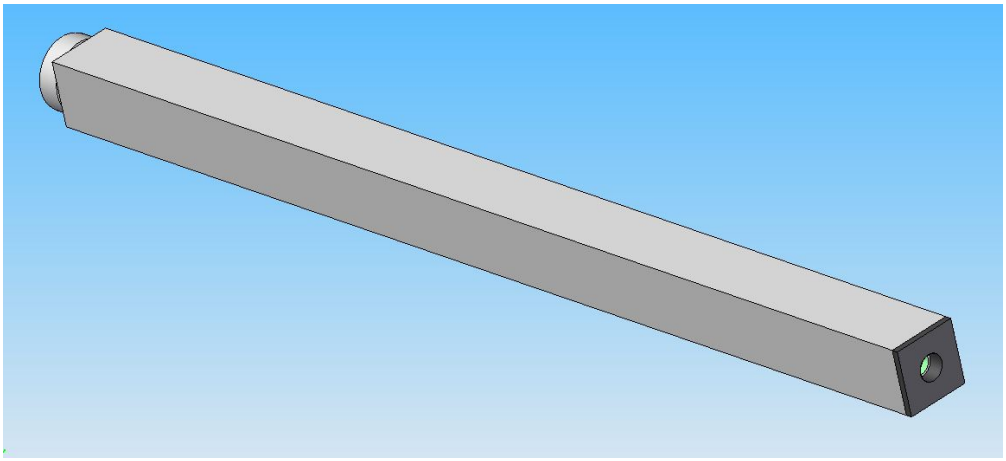
4. Metal Disc
   - Quantity : 2x
   - Type : MSD-17
   - Inner diameter 17 mm, as a counterpart to magnets, not a magnet!

5. Phillips Head Screw M3 x 10
   - Quantity : 2x
   - Material : steel (should be magnetic)

6. Washer M6 x 6.4 x 18 x 1.6
   - Quantity : 1x
   - Material : steel

7. Screw Pan Head Slot M4 x 40
   - Quantity : 1x
   - Material : steel

1.2.5 Rod

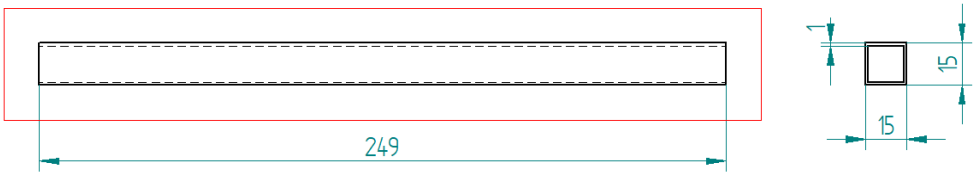


Video

This video shows how to make the Rod

Listing Parts

1. Scare Tube 15 x 15 x 249



2. Insert M4 x 12
3. Magnet

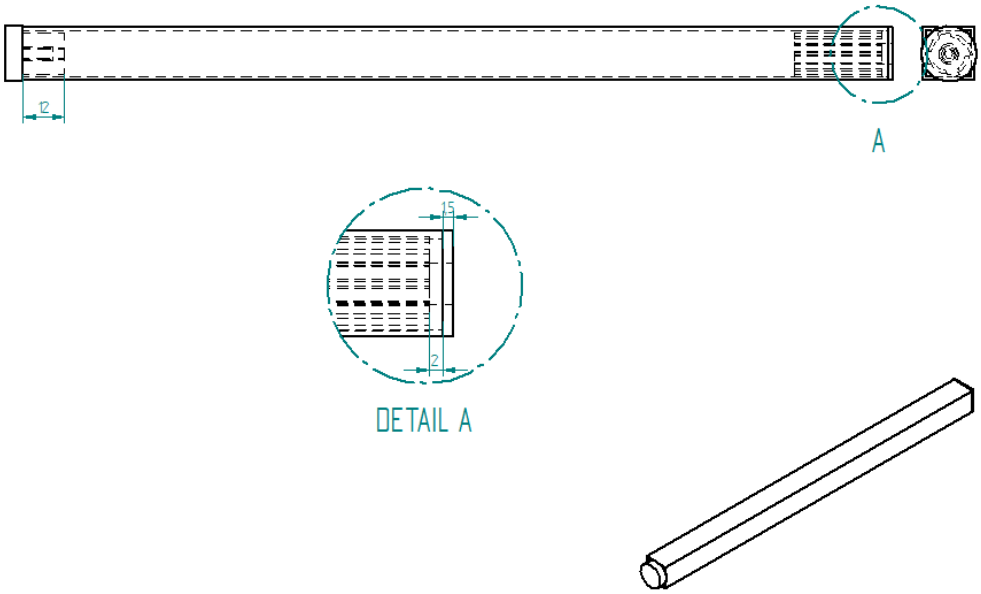
Pot magnet with threaded stud diam. 16mm Thread M4 strength approx. 8kg

- Quantity : 1x
- Type : GTN-16
- Where to buy :

Note: It's important to buy at supermagnete because I bought some in Aliexpress but the strength is lower than those bought at supermagnete

4. Spacer 25 / M6

Manufacturing Process

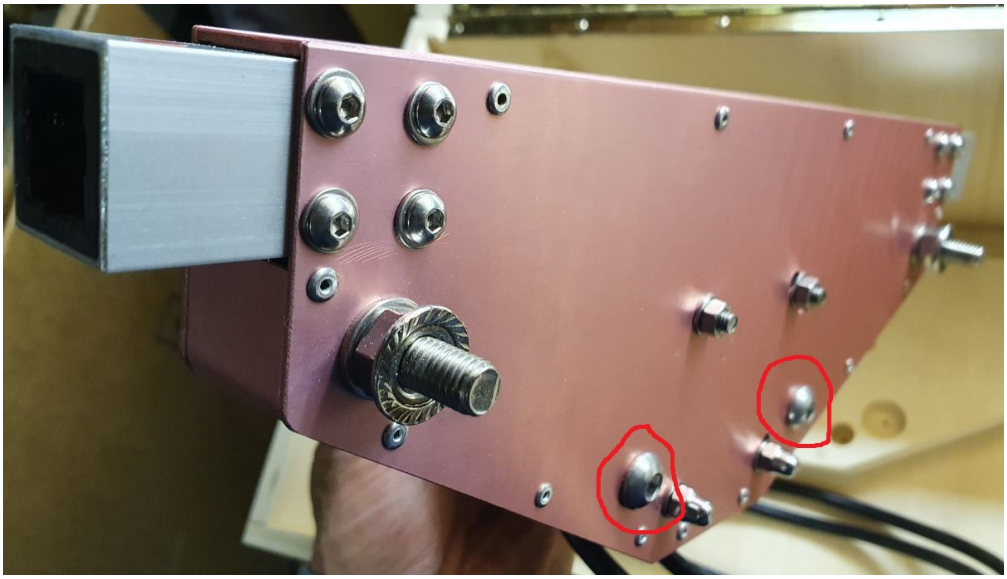


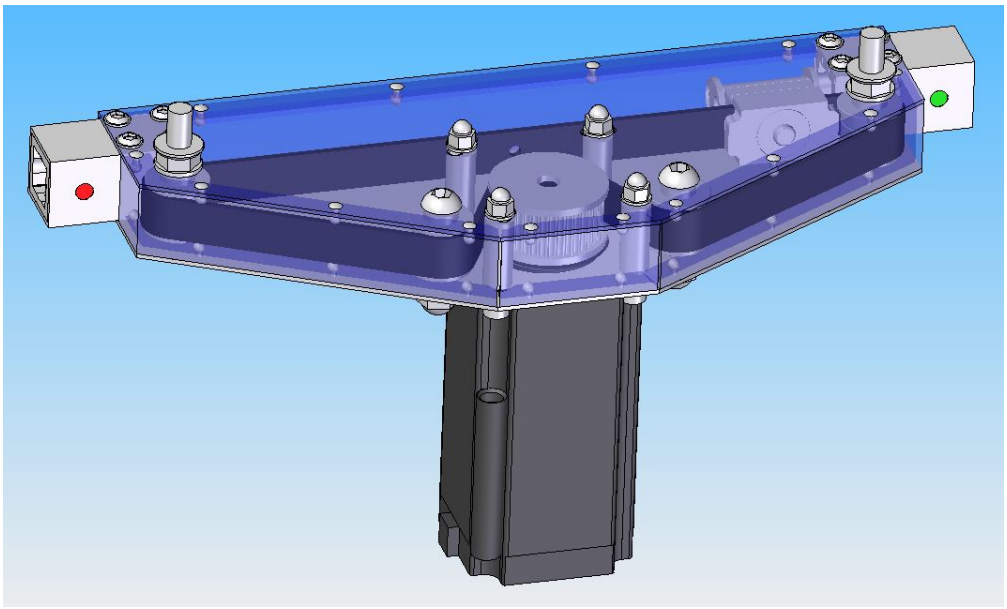
1. Cut the Square Tube 15 x 15 x 249 at length = 249mm
2. Make the the Insert M4 x 12

1.3 Assembly

1.3.1 Video

There is an error occured in this video the bolts (in red) should be mounted like on these images :





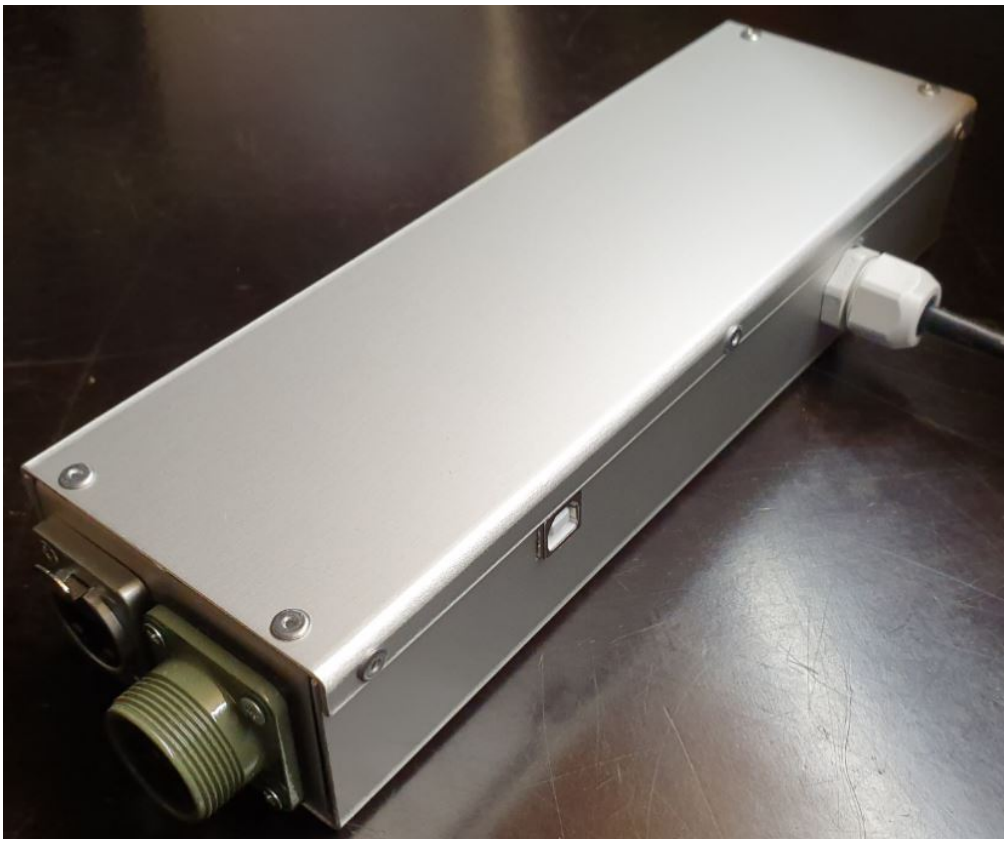
Listing Parts

- Rivets
  - Quantity : 28
  - Supplier : Debrunner

N° d'art.	Prix brut	UE	d1 mm	Type	Long. de serrage mm	Ø perçage mm	d2 mm	d3 mm	k mm	l mm
10100201 793.160.199 pqt	9.80 / 100 pce	100 pce	3.2	TAPD 44 BS	1.6 - 3.2	3.3	6.4	1.8	1.1	6

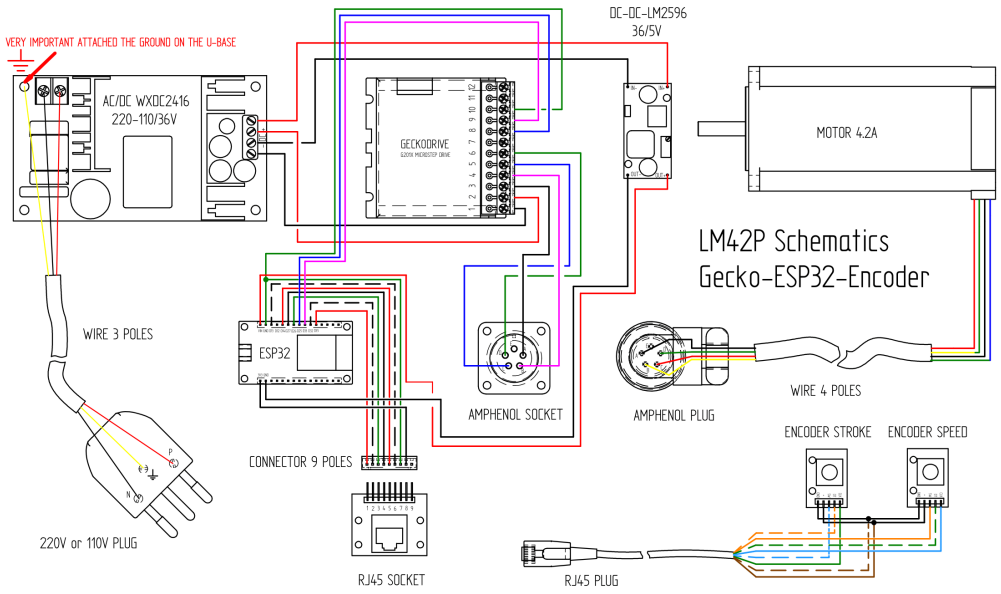
CHAPTER 2

Electrical Part



2.1 Electrical Box with Geckodrive, ESP32 and Encoders Rotary Knobs

2.1.1 Schematic



2.2 Electrical Box with TB6600, Arduino and Analogic Rotary Knobs

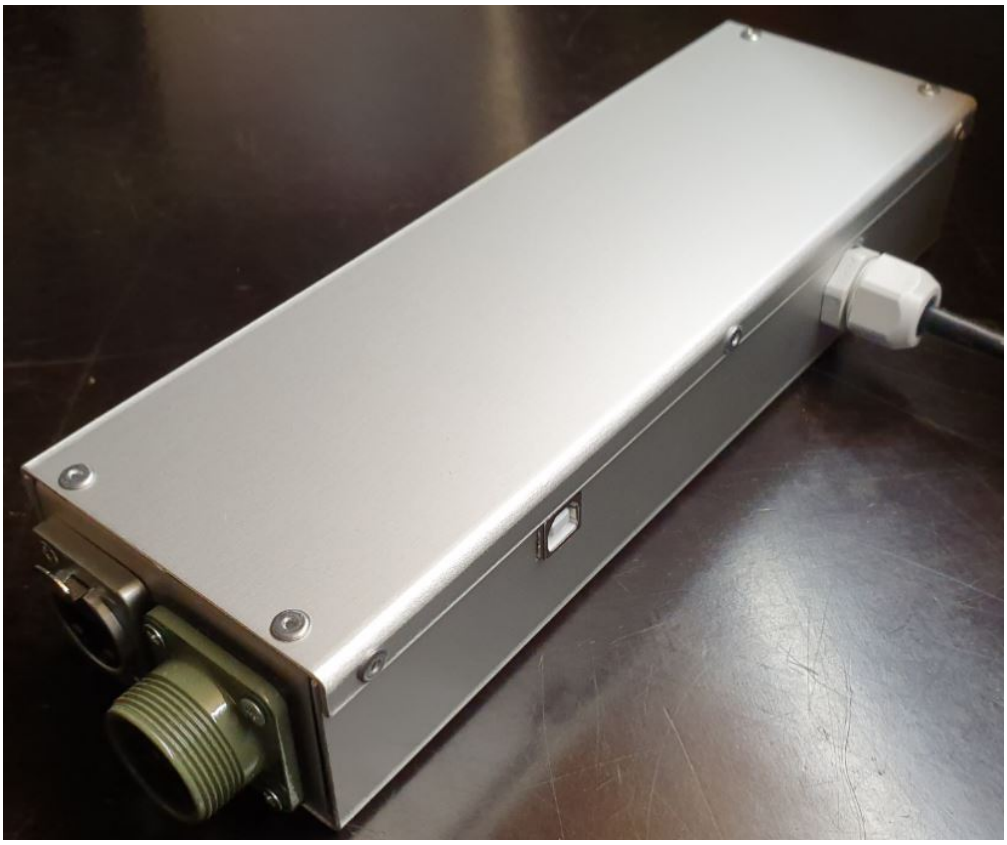
There is another Electrical Box which is about 130$ cheaper but 15-20% slower and 40% less powerful (instead of 29 lbs thrust 17 lbs) The Box is 3D printed.

Driver :


DC9-40VDC
Microstep Driver
SW6
SW5
SW4
SW3
SW2
SW1
High Voltage
Signal
GND
A+
A-
B+
B-
PUL+(+5V)
PUL-(PUL)
DIR+(+5V)
DIR-(DIR)
ENA+(+5V)
ENA-(ENA)
PWR/ALARM

Power 24V, 6A :

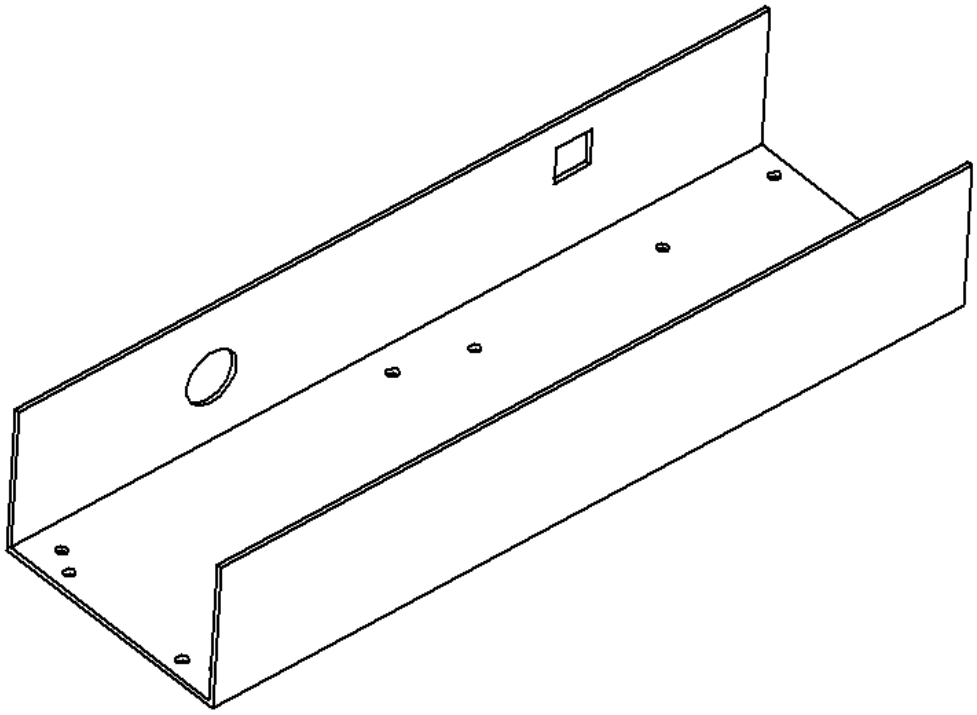
2.3 Electrical Box with Geckodrive, Arduino and Analogic Rotary Knobs



2.3.1 Listing Parts

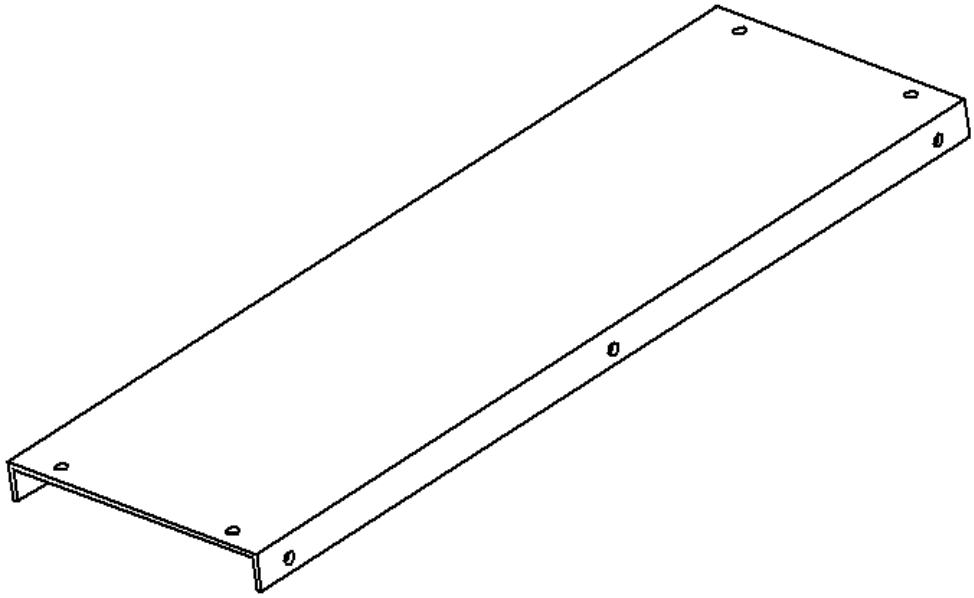
U Base

- Quantity : 1x
- Material : anodized aluminium



U Top

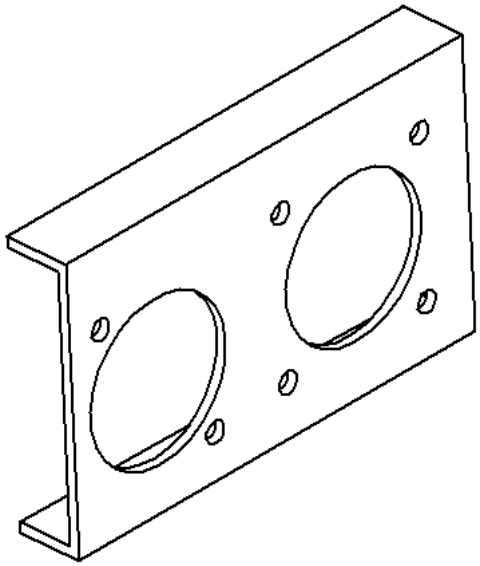
- Quantity : 1x
- Material : anodized aluminium



U Right

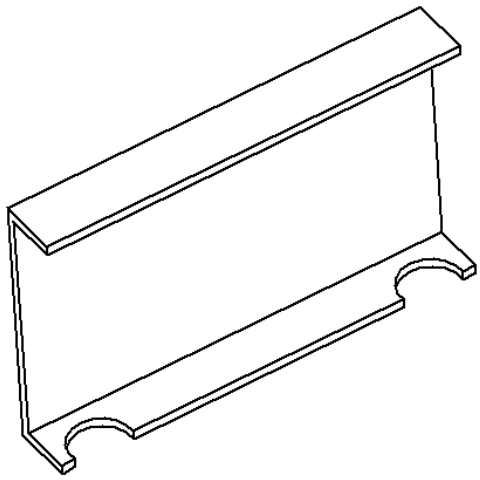
- Quantity : 1x

- Material : anodized aluminium



U Left

- Quantity : 1x
- Material : anodized aluminium



Geckodrive G201X

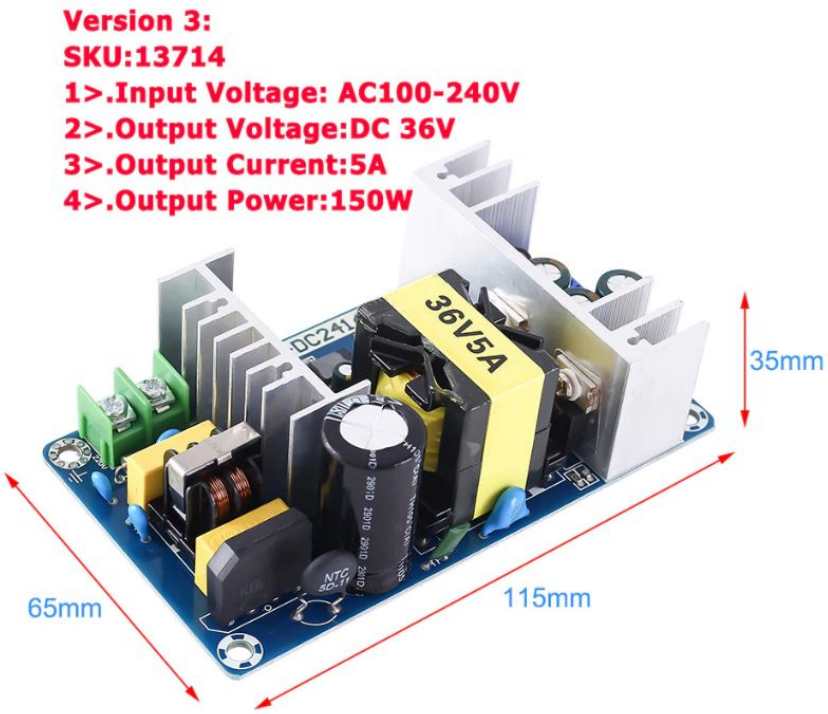
- Quantity : 1x

Arduino Uno

- Quantity : 1x

Power 36V

- Quantity : 1x



Power 36/12V

- Quantity : 1x
- Type : LM2596

Connector Socket 5 pins Female Insert

- Quantity : 1x

Plug RJ45

- Quantity : 1x
- Include : 2x screw M3, 2x nuts M3 and 2x spring washer

Cable Gland

- Quantity : 1x
- Type : M16 4.5-10mm
- Include : 1x nuts M16

Cable Ties

- Quantity : 1x
- Purpose : To be sur that the **Cable Silicone 3 cores** can't be pull out

Cable Silicone 3 Cores

- Length : 1x 2m
- Type : 0.75mm^2
- Note : I used silicone because it's easy to fold.

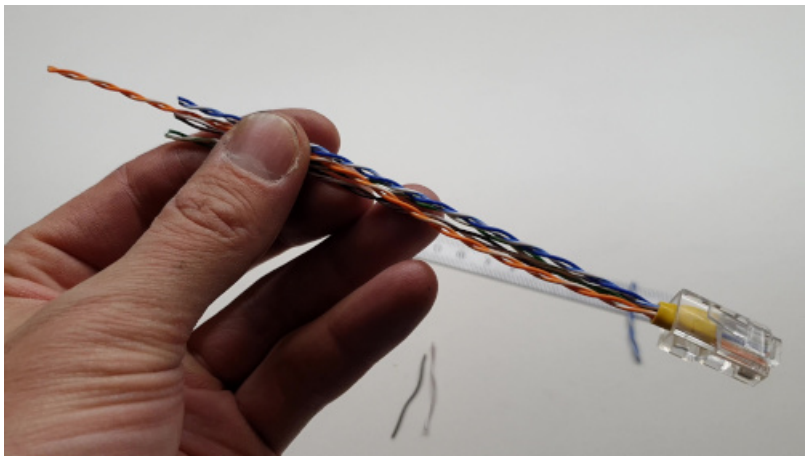
Plug 110 / 220 V

Male PCB PIN Header

- 1x 15 pins
- 1x 10 pins
- 1x 8 pins

Connector RJ45

- Quantity : 1x
- Length : 15cm

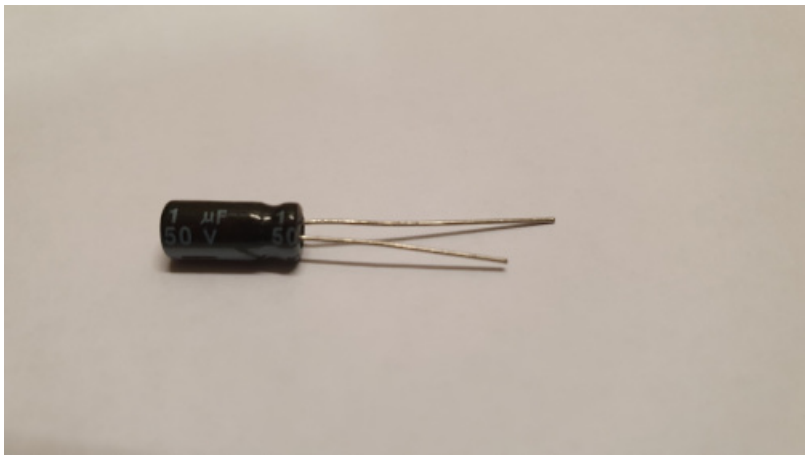


Ground Terminal M3

- Quantity : 1x

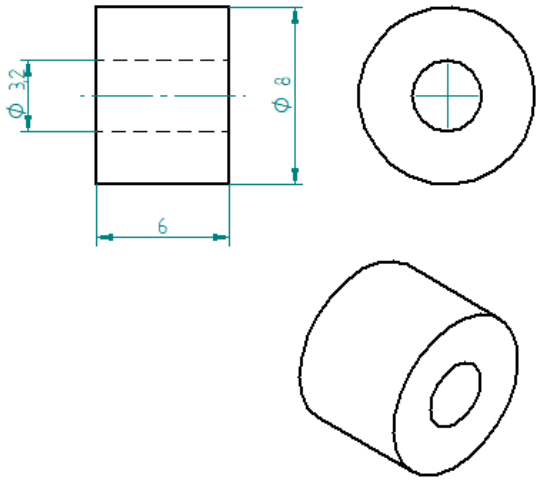
Capacitor Polarized 1uF

- Quantity : 1x
- Purpose : This reduce the noise while reading the speed on the remote-control



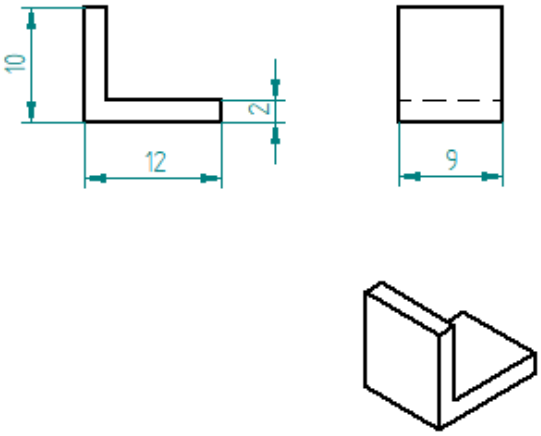
Spacer 6-8-3mm

- Quantity : 4x
- Material : aluminium



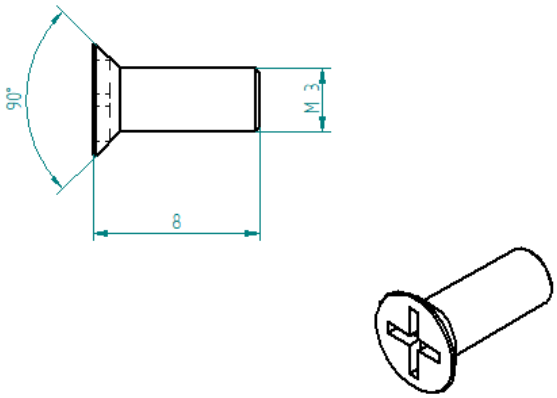
Square 10-12-9mm

- Quantity : 2x
- Material : aluminium
- Purpose : Reinforcement of the fixaton (glue) of the **Arduino Uno** to the **U Base**



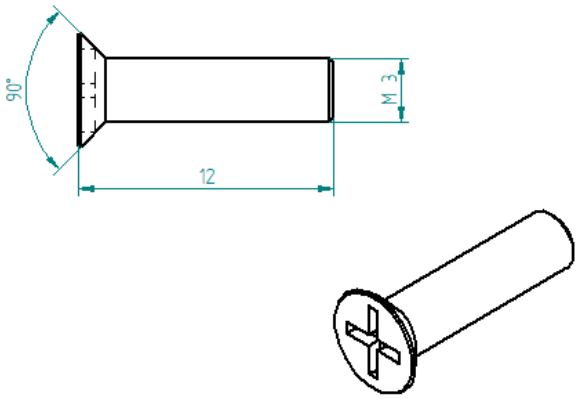
Countersunk Head Screw M3 x 8mm

- Quantity : 4x
- Material : Stainless Steel
- Purpose : To fix **Geckodrive** on the **U Base**



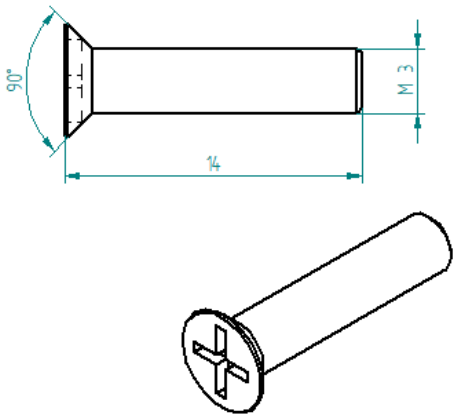
Countersunk Head Screw M3 x 12mm

- Quantity : 4x
- Material : Stainless Steel
- Purpose : To fix the **Connector Socket 5 pins Female Insert** on the **U Right**



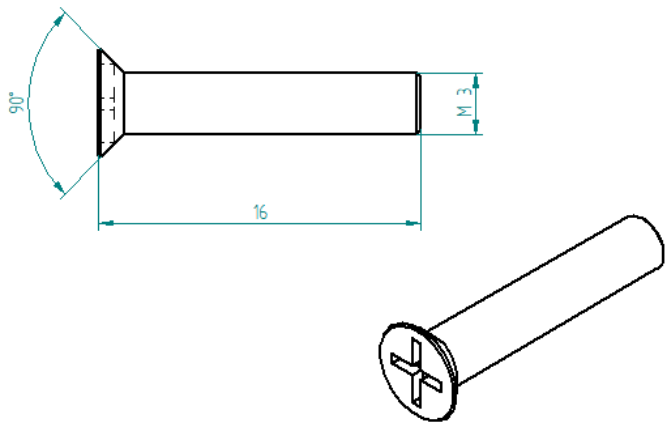
Countersunk Head Screw M3 x 14mm

- Quantity : 3x
- Material : Stainless Steel
- Purpose : To fix the **Power 36V** on the **U Base**



Countersunk Head Screw M3 x 16mm

- Quantity : 1x
- Material : Stainless Steel
- Purpose : To fix the **Power 36V** also the **Ground Terminal** on the **U Base**



Washer M3

- Quantity : 12x

Nuts M3

- Quantity : 12x

Rivet

- Quantity : 14x
- Type : diameter 3mm, length 6mm
- Material : Stainless Steel

Glue Silicone

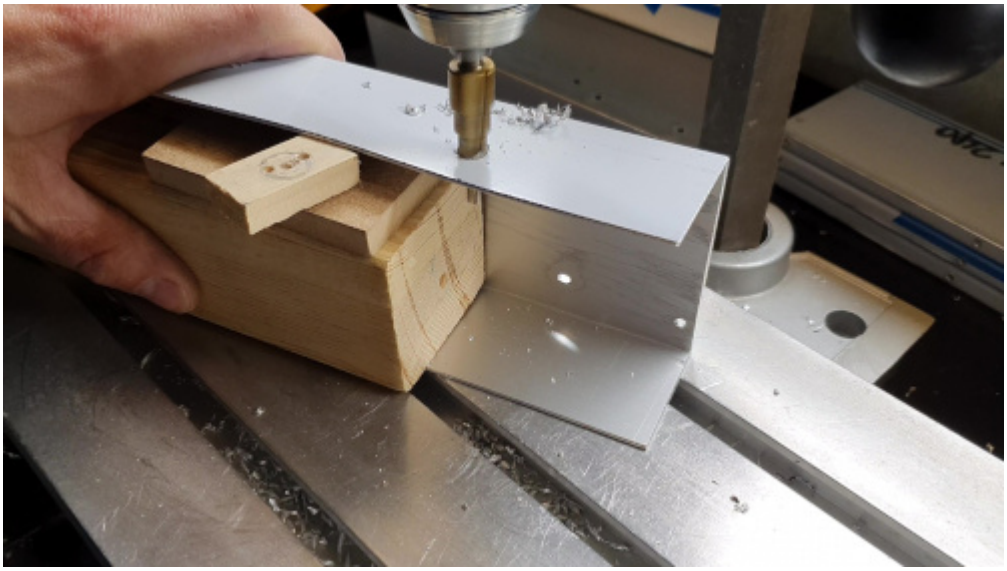
- Quantity : 1x
- Type : Polyflex 444
- Purpose : To glue the **Arduino Uno** to the **U Base** using the **Square 10-12-9mm** to fix stronger

Threadlocker Glue

- Quantity : 1x
- Type : Loctite 243
- Purpose : To glue the **Nuts M3**

2.3.2 Operation Plan

Make The Sheets Parts



Make the **U Base**, **U Right** and the **U Left** following this **Video**, see also the drawings below :

Drawings :

Material : aluminium

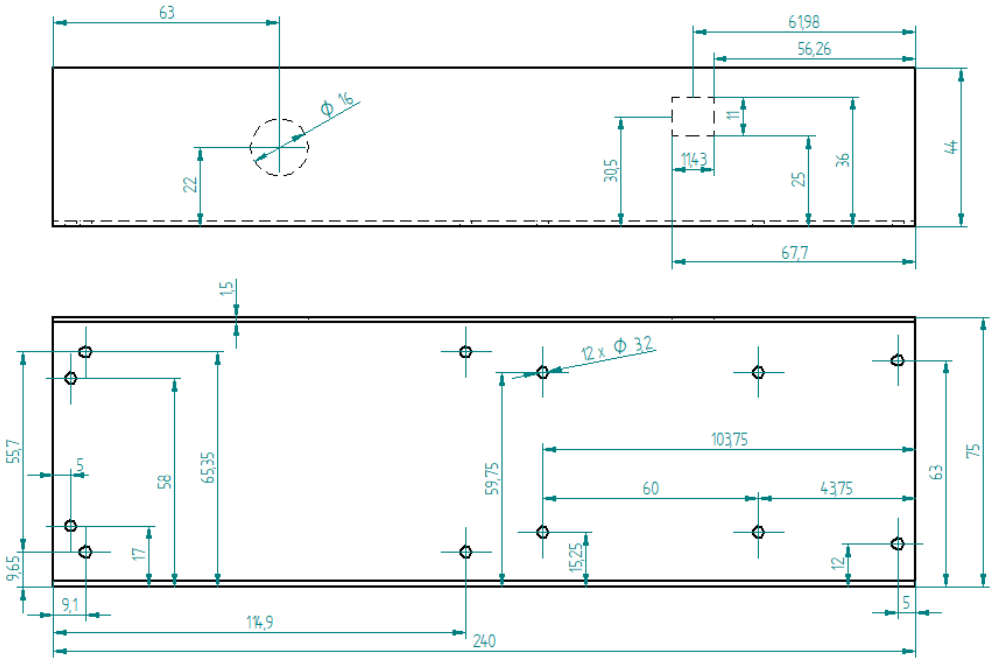


Fig. 1: U Base

Make the Spacer 6-8-3mm

See the following video :

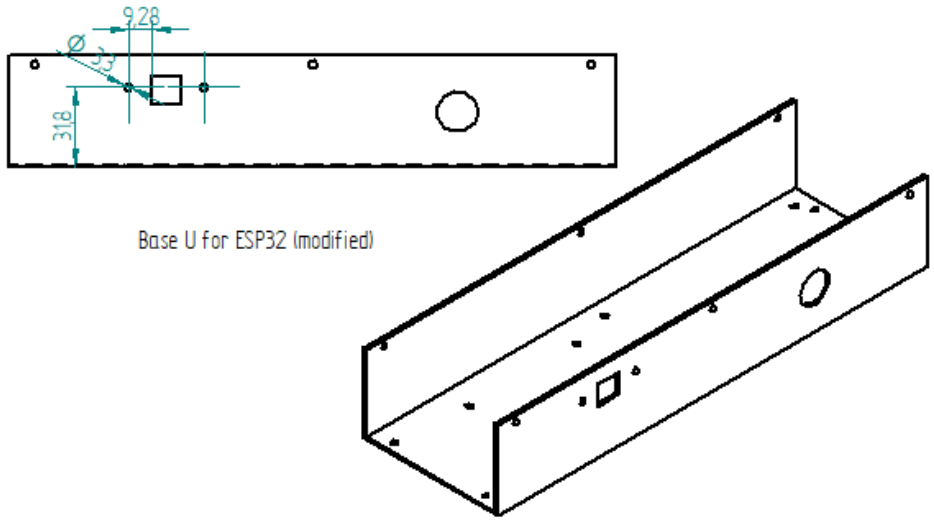


Fig. 2: U Base ESP (modified)

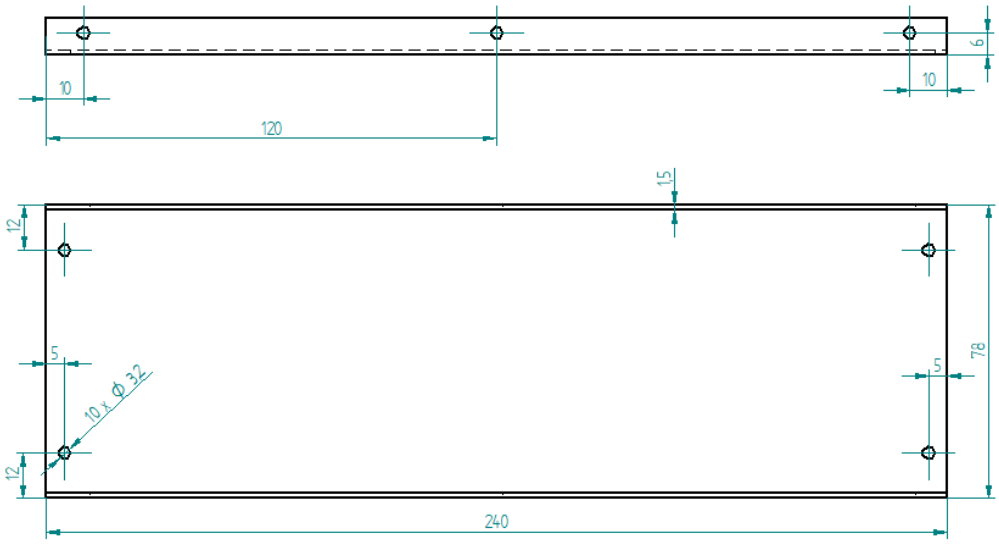


Fig. 3: U Top

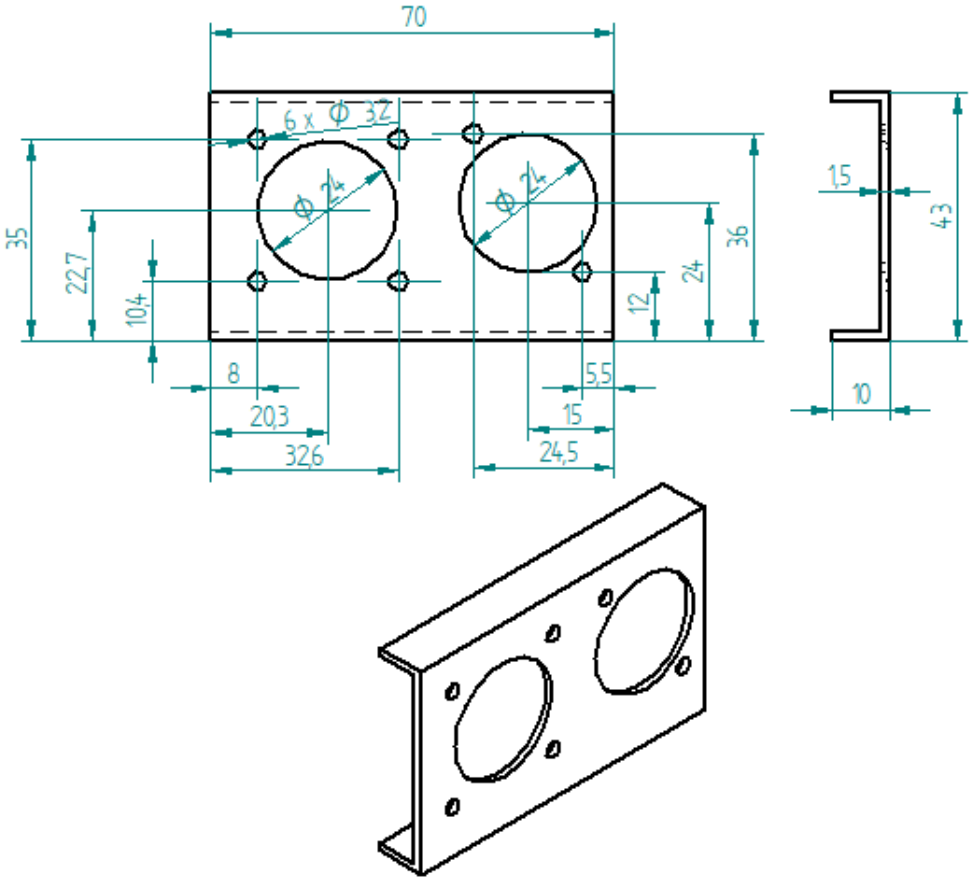


Fig. 4: U Right

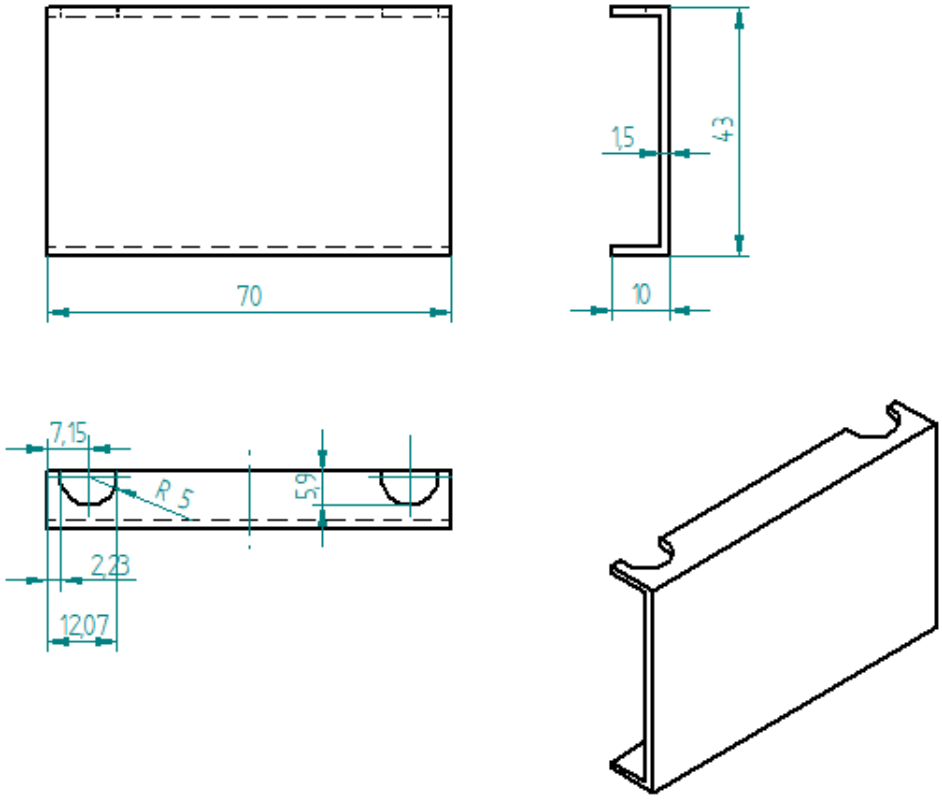


Fig. 5: U Left

Drawing :

- Quantity : 4x
- Material : aluminium

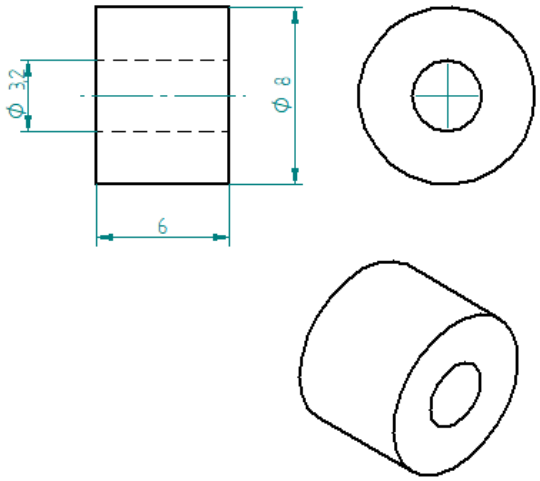


Fig. 6: Spacer 6-8-3mm

Make the Square 10-12-9mm

See the following video :

Drawing :

- Quantity : 2x
- Material : aluminium

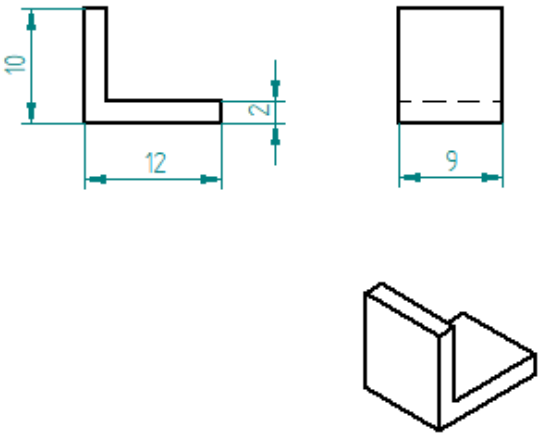
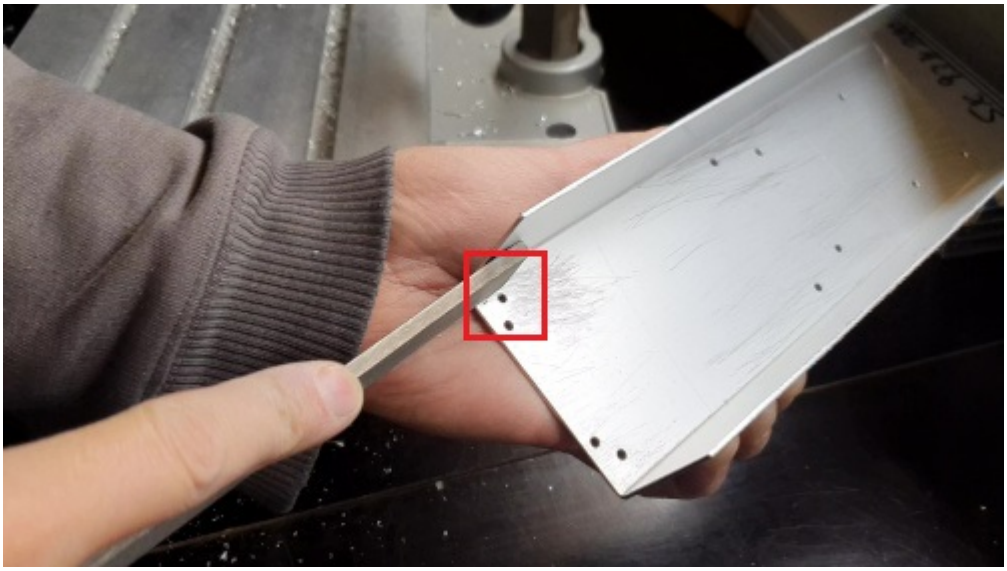


Fig. 7: Square 10-12-9mm

Scrape the surface on the U Base

File the bottom of the **U Base** (only the part where the red framed hole is) so that the grounding contact faces well. This operation is not necessary if you are using non anodized sheets.

Note: The anodized surfaces are not conductive.



See Video :

Control Power 36V

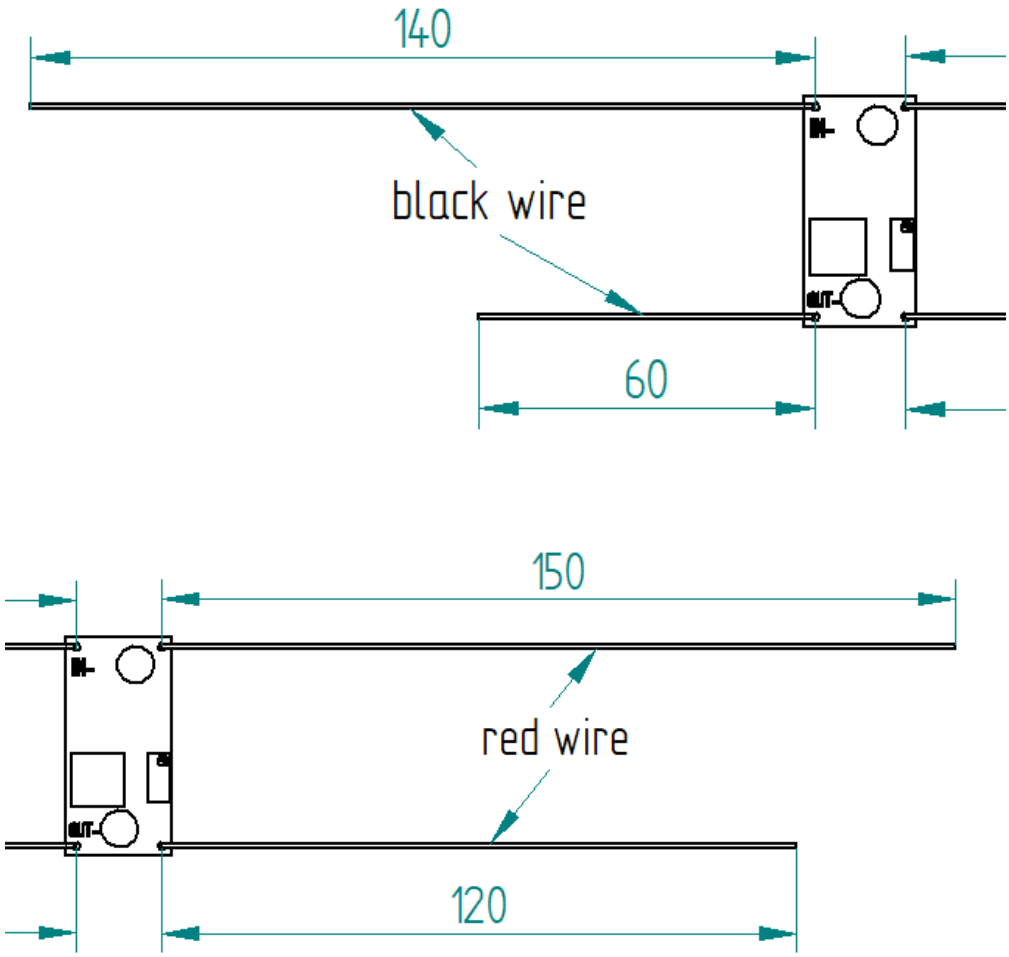
Control the voltage of the output of the Power 36V with a voltmeter. It should be 36V.

Adjust voltage Power 36/12V

Materials:

- 1 wire 0.5mm^2 red length = 120mm
- 1 wire 0.5mm^2 red length = 150mm
- 1 wire 0.5mm^2 black length = 140mm
- 1 wire 0.5mm^2 black length = 60mm

1) Sold the wire



2) Wire the Power 36/12V

- Connect the Power 36/12V IN to the Power 36V OUT
- Connect the voltmeter to Power 36/12V OUT

3) Adjust the voltage

   With a *Screwdriver 0*, adjust the voltage to 12V

Fix the Power Cable to the Electrical Box

See Video :

1. Strip the **Cable Silicone 3 Cores** at 10cm
2. Fix the **Cable Gland** to the **U Base**
3. Tighten **Cable Gland**
4. Tighten the **Cable Ties** and cut it with a *Cutting Pliers*
5. Tighten the **Ground Terminal M3** on the ground wire (yellow)
6. Tighten the Phase and Neutre to **Power 36V** “IN”

Set Geckodrive current limit

This operation limits the current so that the motor will not overheat. I tested with 5A and it looks like the motor is a bit overheaten. But thats ok if the machine is used less than about 10min. To be sure the motor is not overheaten I recommand to set the current at 4.2Amps.

- For G203V :

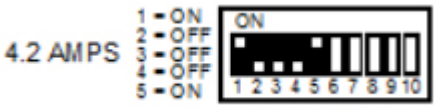
  Connect a resistor of 120kOhms between pin 11 and 12 of the GECKODRIVE. This will limit MOTOR current by 5A. (for 4.2A please refer to datasheet)

- For G201X :
  - if machine used for less than 10min set to 5A

    set the switches like the following figure

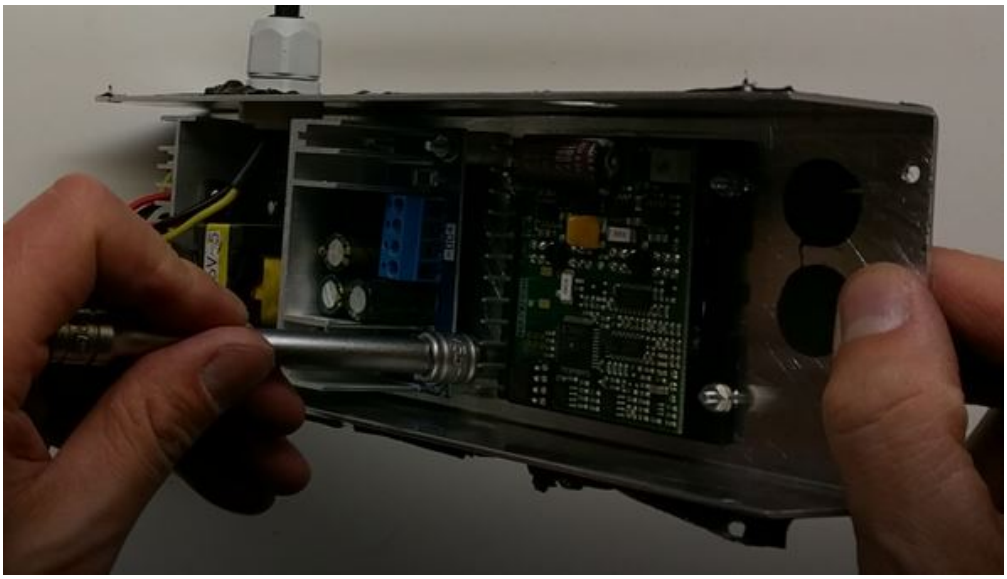
  - if machine used more than 10min set to 4.2A

    set the switches like the following figure



Screw the Power 36V and Geckodrive on the U Base

Note: Use **Threadlocker Glue**.

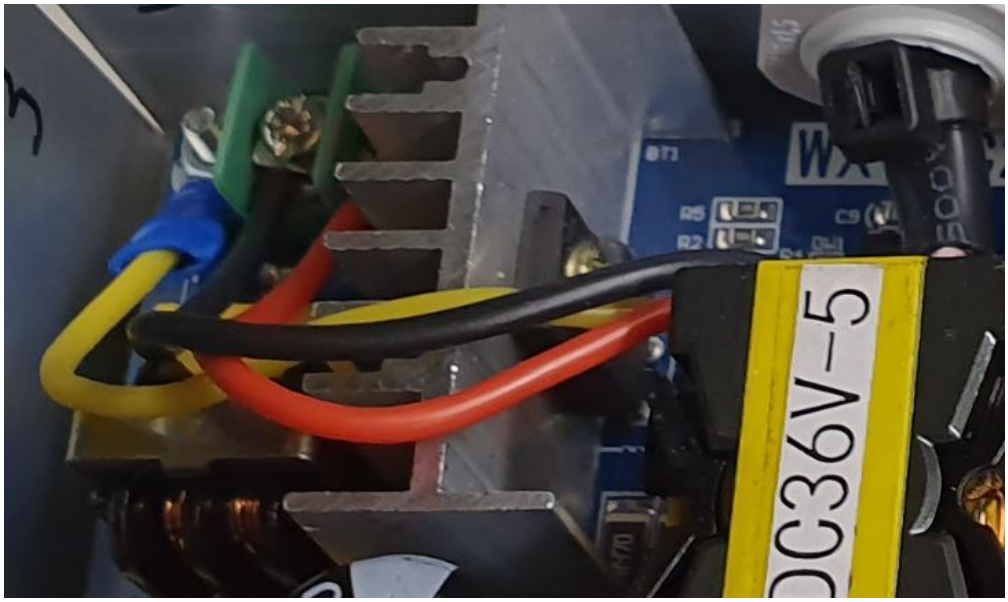


- For Power 36V :
  1. **Fix three corners of the Power 36V by using :**

- 3x **Spacer 6-8-3mm**
- 3x **Countersunk Head Screw M3 x 14mm**
- 3x **Washer M3**
- 3x **Nuts M3**

2. **Fix the last corner** [The ground to the U Base by using :]
   - 1x **Countersunk Head Screw M3 x 16mm**
   - 1x **Washer M3**
   - 1x **Nuts M3**
   - 1x **Ground Terminal M3** (the ground on the **Cable Silicone 3 Cores**

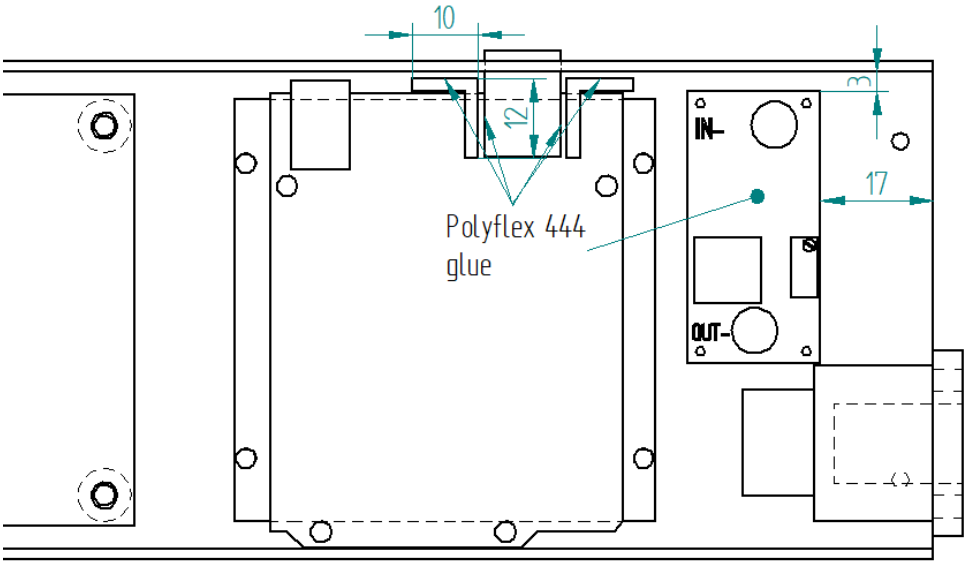
Power 36V	U Base
GROUND	screw with *Spacer*

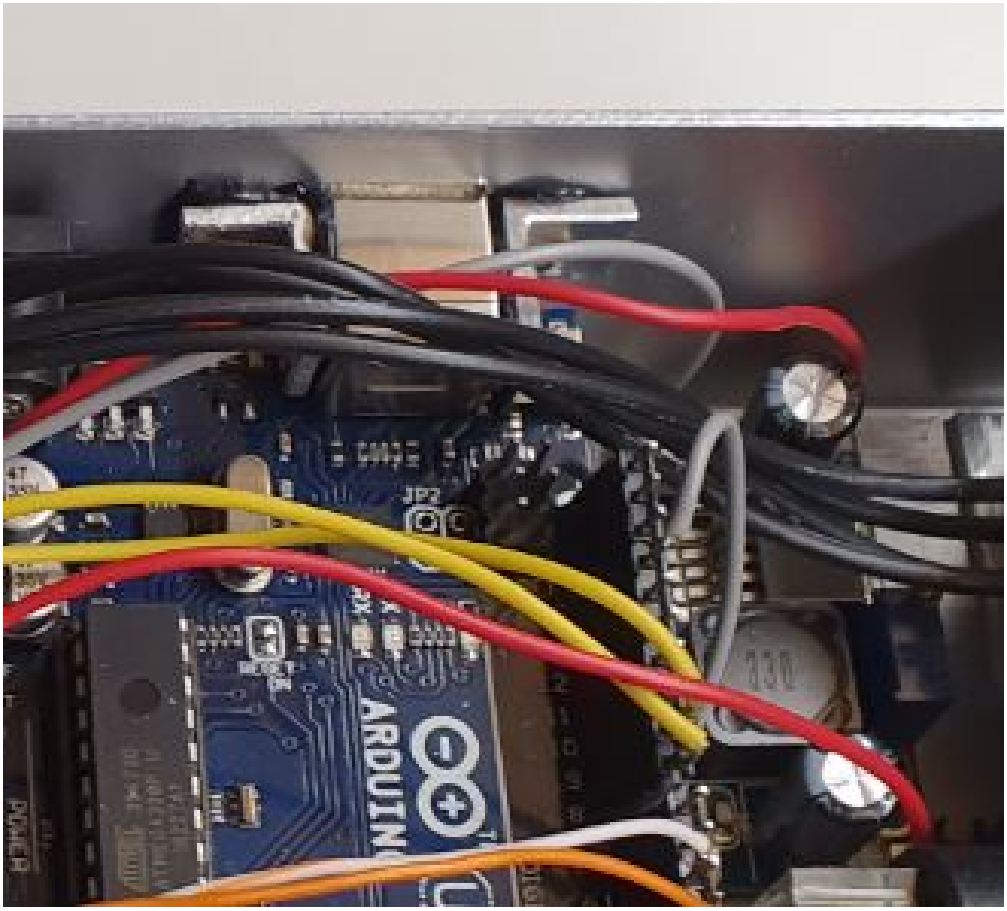


- For Geckodrive use :
  - 4x **Head Screw M3 x 8mm**
  - 4x **Washer M3**
  - 4x **Nuts M3**

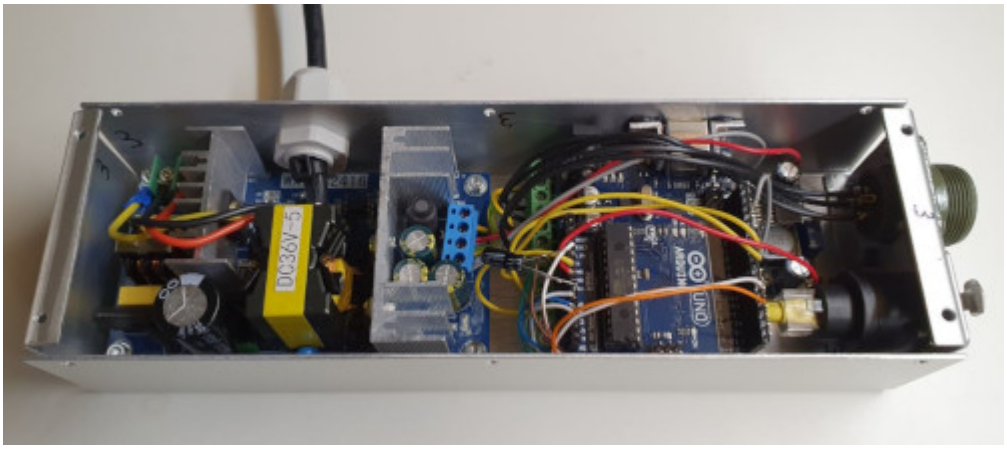
Glue the Power 36/12V and Arduino Uno

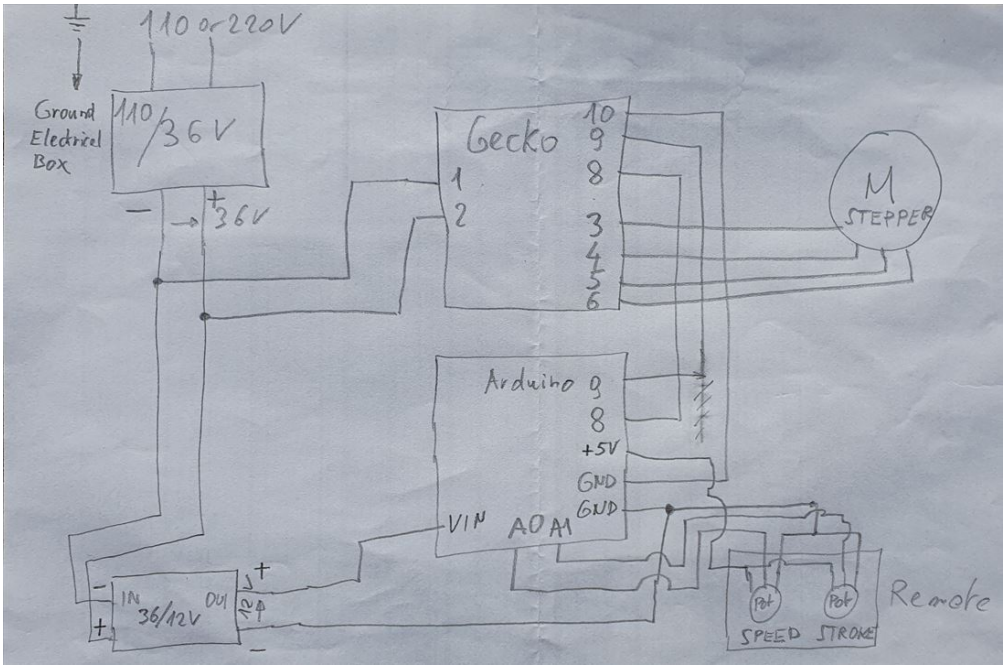
Glue the **Arduino Uno** at the **U Base** with **Glue Silicone** and **Square 10-12-9mm** and the **Power 36/12V**.


10
12
3
17
IN-
OUT-
Polyflex 444
glue



Wiring





See video :

1. Sold the wires to the **Male PCB PIN Header** (15, 10, 8 pins), except the POWER 36/12V OUT+

   Use two clamps this help to sold the **Male PCB PIN Header**

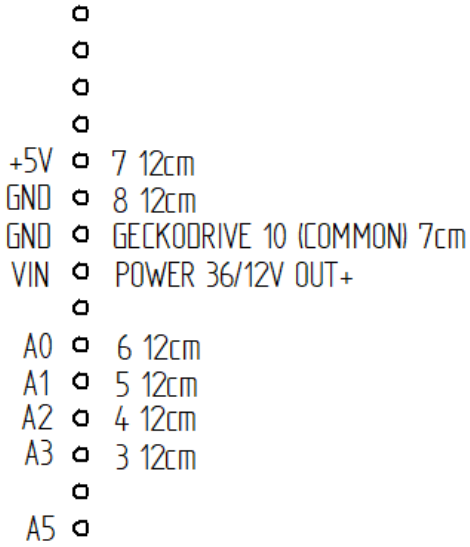


Fig. 8: Male PCB PIN Header 15 pins

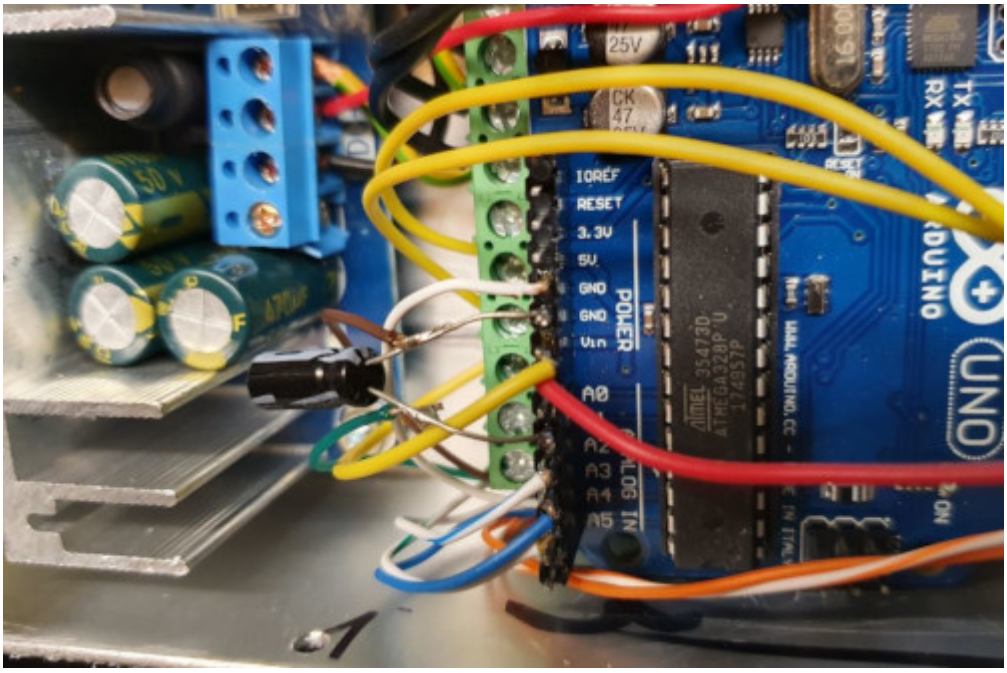
GECKODRIVE	ARDUINO	Cable Length
8 (DIR)	PIN 8	11cm
9 (STEP)	PIN 9	11cm
10 (COMMON)	GND	7cm

Fig. 9: Male PCB PIN Header 10 and 8 pins

POWER 36/12V	ARDUINO	Cable Length
OUT-	GND	6cm
OUT+	VIN	12cm

ARDUINO	RJ45 cable (inside Box)	Cable Length
A0	6 sold capacitor +	12cm
A1	5	“
A2	4	“
A3	3	“
~3	2	15cm
~5	1	“
GND	8 sold capacitor -	12cm
+5V	7	“

2. Sold the Capacitor between A0 and GND (8) see folowing picture



3. Connect :

Power 36V	GECKODRIVE	Cable Length
-DC	1 (POWER GND)	6cm
D+	2 (18 TO 80 VDC)	“

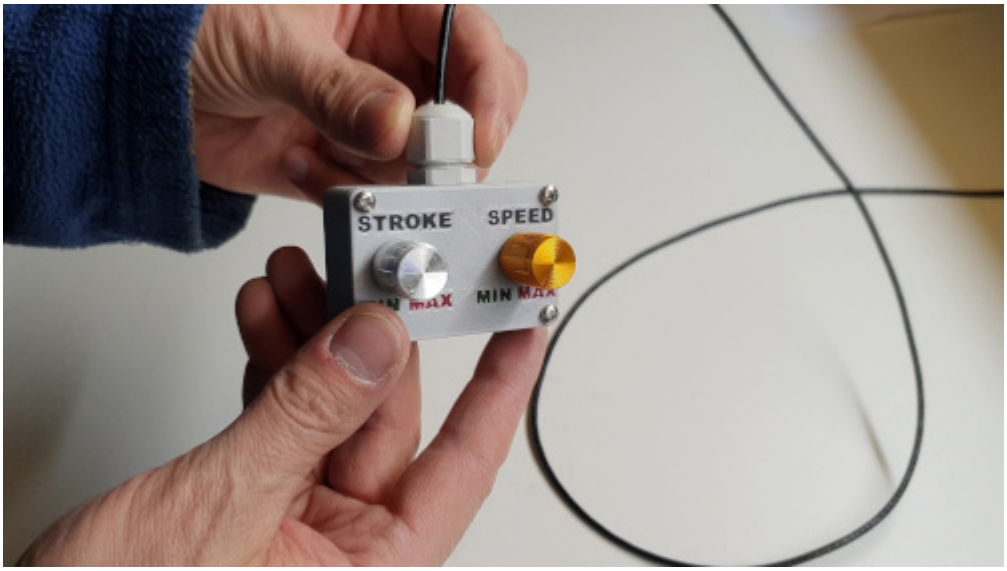
Power 36V	POWER 36/12V	Cable Length
-DC	IN-	14cm
DC+	IN+	15cm

GECKODRIVE	Female Connector (MOTOR)	Cable Length
3 (WINDING A)	A	15cm
4 (WINDING not A)	B	“
5 (WINDING B)	C	“
6 (WINDING not B)	D	“

Upload the programm to Arduino Uno and final control

1. Plug the PC using an USB cable to the Electrical-Box
2. Upload the programm (soon available)
3. Control if every thing is ok (the ***Machine**, **Remote Control**, **Stepper Motor** and the **Rod** should be done). Do all these steps showed in that video for the test :
4. Cover with **U Top** and **Rivet**

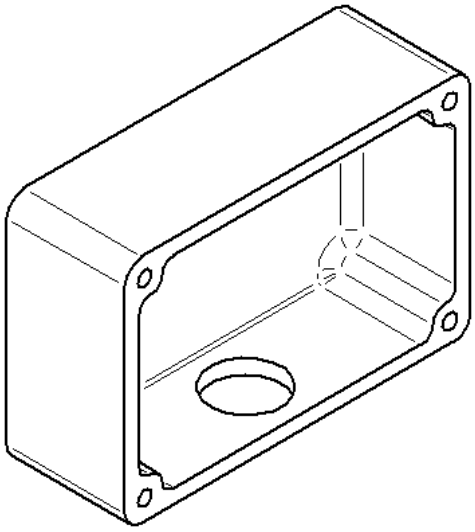
2.4 Remote Control



2.4.1 Listing Parts

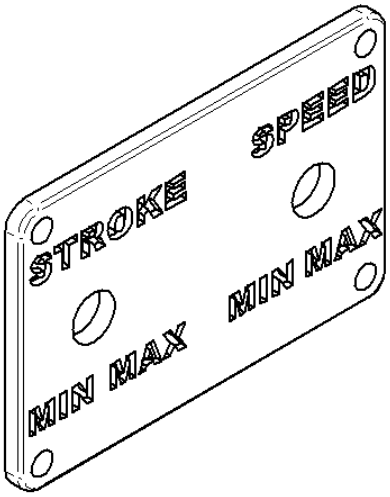
Box

- Quantity : 1x
- 3D Printed in PLA



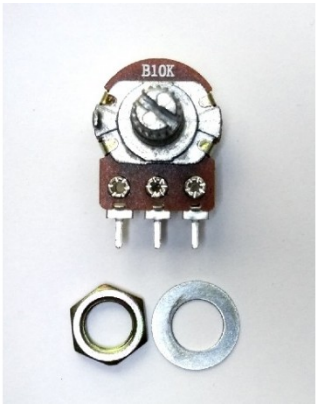
Cover

- Quantity : 1x
- 3D Printed in PLA



Potentiometer

- Quantity : 2x
- Type : 10k



Potentiommeter Knobs

- Quantity : 1x Grey (stroke), 1x Gold (speed)

RJ45 Cable

- Quantity : 1x
- Length : 2.5m
- Type : Ultrafine

Cable Gland M12

- Quantity : 1x
- Type : M12 3-6mm

Epoxy

- Resin and Hardener

Pigment Epoxy

- Quantity : 1x color black, 1x color red, 1x color green

Cable Ties

- Quantity : 1x
- Purpose : To be sur that the **RJ45 Cable** can't be pull out

Screw

- Quantity : 4x
- Diameter : 2.9mm
- Length : 13mm
- Stainless steel

2.4.2 Operation Plan

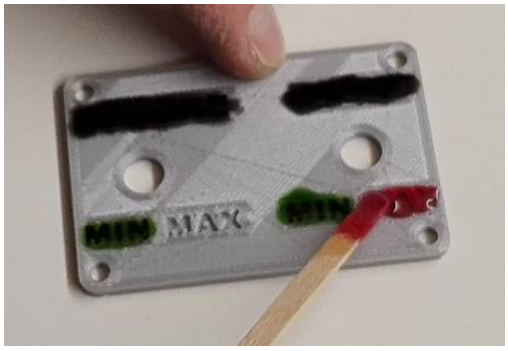
See the video :

Make the Box

Print it in PLA. Infill = 50% File : soon available...

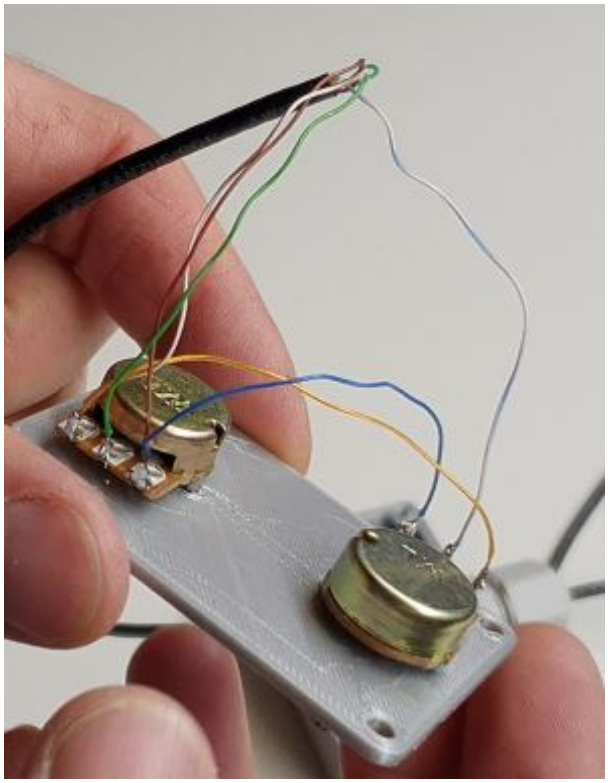
Make the Cover

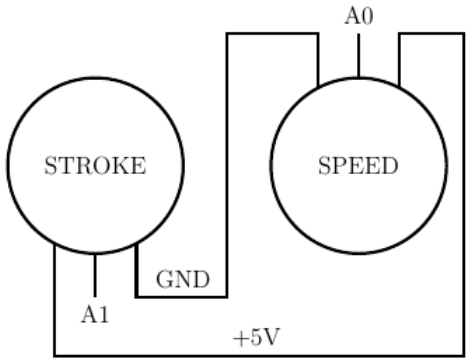
1. Print it in PLA. Infill = 50% File : soon available...
2. Prepare the **Epoxy** with black pigment, red pigment and green pigment
3. Fill with pigmented Epoxy the inscriptions on the **Cover**



4. Cure the **Epoxy** then sand it very thin (60, 240, 400)
5. Place the **Potentiometer** (2x) and and tighten them

Wiring the Remote Control





Remote-Control	RJ45 cable	Wire Color
A0	6	G
A1	5	b
GND	8	BR
+5V	7	br

1. Strip the **RJ45 Cable** at 7cm
2. Cut the unused wire
3. Use the cutted unused wire to make the bridge between GRD and +5V inside the Remote Control

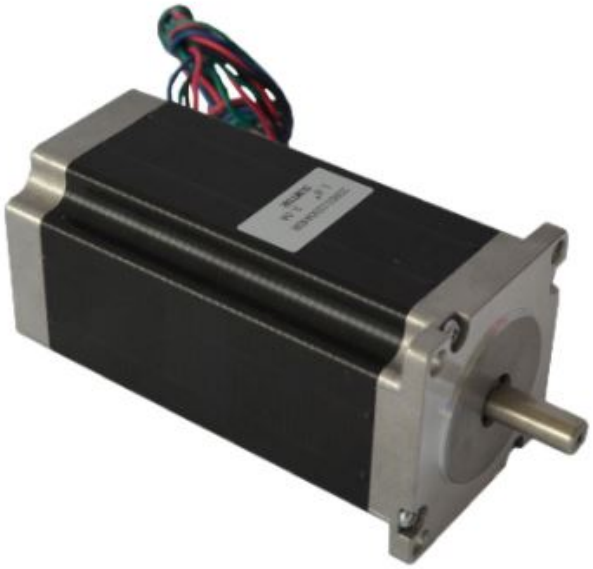
2.5 Motor

This section shows how to fix the **Cable Silicone 4 Cores** to the **Connector Plug 5 pins Male Insert** and the **Motor**

2.5.1 Listing Parts

Motor

- Quantity : 1x
- Type : Stepper Motor NEMA 23 23HS11240
- Length : 112mm
- 4.2A



Connector Plug 5 pins Male Insert

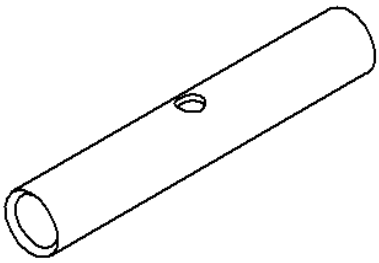
- Quantity : 1x

Cable Silicone 4 Cores

- Quantity : 2.2m
- 0.75 mm^2

Tube Cable Holder

- Quantity : 1x
- Material : Aluminium



Heat Shrink Tube

- Quantity : 4x - Length : 13mm
- Quantity : 1x - Length : 37mm

Screw Hex Head Allen M5 x 40

If the **Motor** isn't fixed yet on the **Machine**, then 1x Screw is needed.

- Quantity : 1x
- Size : M5 x 40
- Type : Stainless Allen Bolt Socket Cap Screw Hex Head Allen Key DIN912

Glue Silicone

- Quantity : 1x
- Type : Polyflex 444

Cable Ties

- Quantity : 1x

2.5.2 Connect the Connector Plug 5 pins Male Insert

Operation Plan

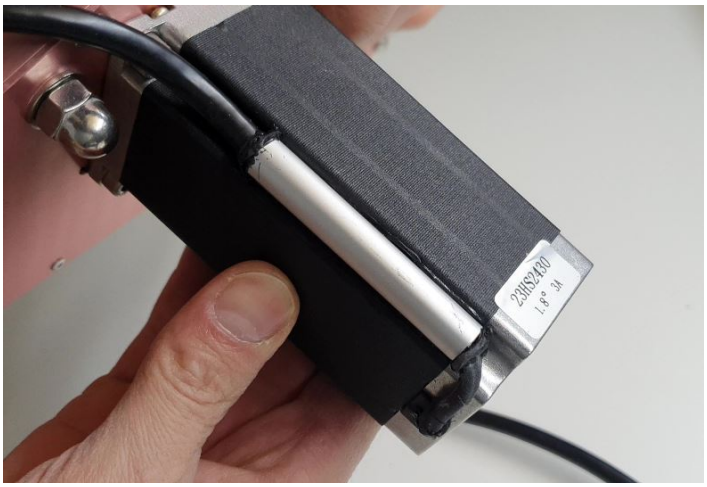
See the video :

Wiring

1. strip the cable at 3cm
2. strip the 4 wires and solder the wires
3. 5cm from the edge roll up the 10-layers adhesive tape
4. pass the cable and solder the wires in the plug in counter-clockwise order
    - black (A)
    - yellow (B)
    - red (C)
    - green (D)

5. assemble the plug and tighten the flange

2.5.3 Connect and fix the Cable Silicone 4 Cores to the Motor



Operation Plan

See the video :

1. Make the **Tube Cable Holder**

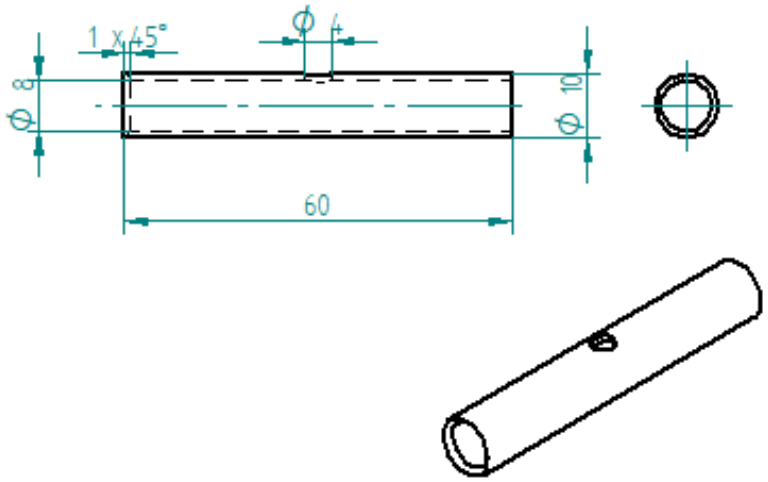


Fig. 10: Tube Cable Holder (material aluminium anodized)

2. Unpacking and motor control. Plug the 4 wires on an **Electrical Box** and make it turn. This is important before making the following steps.
3. There are two kinds of motor a 3A and a 4.2A each motor has their own colors of wire. Depend on the Amps of the motor you use, cut the wire as follow :
    1. Motor 3A :

- red wire 47mm
- yellow 57mm
- blue 67mm
- green 77mm

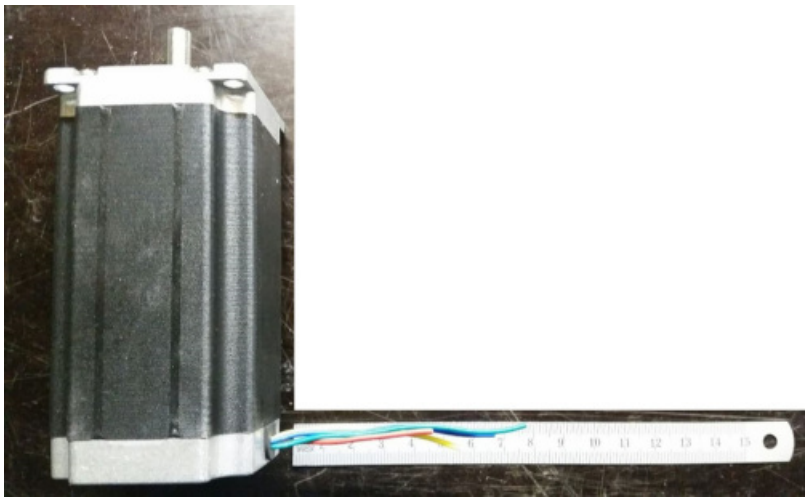


Fig. 11: 3A motor with wires cutted

2. Motor 4.2A :
    - red 47mm
    - green 57mm
    - black 67mm
    - blue 77mm

4. Stripping, twisting, tinning the motor wires to 5mm
5. Cut the **Heat Shrink Tube** to 37mm length and tighten it with industrial foehn. The red wire should protrude about 5mm..
6. Cut the **Cable Silicone 4 Cores** at 2.2m
7. Strip the **Cable Silicone 4 Cores** to 45mm.
8. Cut the wires on the **Cable Silicone 4 Cores** as follow: - red 40mm - yellow 30mm - black 20mm - green 10mm
9. Strip, twist, tin-plate the cable wires to 5mm.
10. Cut 4x **Heat Shrink Tube** at 13mm
11. Put 13mm **Heat Shrink Tube + Tube Cable Holder** (pay attention to direction the chamfer 1x45) and solden
12. Degrease **Tube Cable Holder**, **Cable Silicone 4 Cores** and **Motor** with acetone

> **Warning:** If the **Motor** isn't fixed on the machine yet, then don't forget to put the screw in the motor hole (see picture below)

13. Inject the **Glue Silicone** through the 4mm hole diam. until it comes out of both sides of the **Tube Cable Holder**; take the surplus and apply it to the **Motor** on the groove where the **Screw Hex Head Allen M5 x 40** has been

placed; turn the **Tube Cable Holder** so that the injection hole is against the **Motor** and is not visible; put a **Cable Ties** on the **Cable Silicone 4 Cores** , put the **Tube Cable Holder** on with the clamp see following picture

if necessary, inject at the end of the **Tube Cable Holder** where the chamfer is located and put some **Glue Silicone** on the **Motor** if you see the wires that protrude a little beyond the **Heat Shrink Tube**

14. Allow to harden; clean and remove the beads on **Tube Cable Holder**.

CHAPTER 3

Code

Here is the Arduino code. You can download the latest and the previous versions from here : . I'm not a professional programmer. The code is a bit messy/dirty sorry about that. At the begining I tried to use AccelStepper library for Arduino but couldn't get it work well. (couldn't reach an appropriate speeed I don't know why) so I remove AccelStepper library and tried to do something and it worked not badly. The code are named: **CodeGeckoVx.ino**. But after that a few person ask me to remove the jump of speed at 90% of the speed knob and to improve the moves when heavy dildos are used around 2kg, I made another version using the library accelStepper. The speed isn't so high as the original one (CodeGeckoVx.ino) but it afford more reliability when the move change direction (espacielly with big loads). That code is called **CodeGeckoAccelStepVx.ino**. Find the latest code underneath. Sorry the comments in the code are in french, I'll translate them soon!

Maybe this video can help to understand the code

3.1 CodeGeckoV5.ino

```
// CodeGeckoV5.ino
// V4->V5 course_min a été mis à 296 au lieu de 380 permet peut-être d'avoir un␣
↪meilleure vibro
// V3 -> V4 Elimine le full stroke quand on débranche la Télécommande en pleine␣
↪action.
// Comme le fait de lire la vitesse du potentiomètre prend relativement beaucoup de␣
↪temps,
// cela réduit la vitesse de pulsation de la pin 9. Pour parer à ceci le programme a␣
↪été adapté
// pour lire la vitesse tous les 300 itérations à basse vitesse.
// Lorsque la vitesse du potentiomètre dépasse v_max_inv qui est la vitesse maximale␣
↪à laquelle
// le moteur ne décroche pas à l'inversion du sens de rotation, des boucles d
↪'accélérations et de
// décélération sont requises pour ne pas faire décrocher le moteur lors des␣
↪inversions de sens de
```

(continues on next page)

(continued from previous page)

```
// rotations. (avec ceci, il est possible de diminuer le délais entre pulse à haute␣
↪vitesse ce
// qui augmente la vitesse) Comme la vitesse est rapide la vitesse est alors lue␣
↪après un cycle
// complet de va-et-vient. Pour simplifier, la course n'est lue qu'après un cycle␣
↪complet de va-et-vient.
// Un autre point important est de lire la vitesse souvent à basse vitesse :
// par exemple si la vitesse est au mini et la course est grande, alors cela prendra␣
↪une éternité
// pour que le va-et-vient se termine jusqu'à la prochaine lecture de vitesse.
// L'accélération et la décélération est linéaire (après un cycle de pulse, le délais␣
↪est soustrait pour
// l'accélération ou additioné de 1 pour la décélération).

#define step_pin 9          // Pin 9 connected to Steps pin on ST-M5045 ou ... c'est␣
↪la pin du pulse
                            // (un pulse c'est un pas en mode full
                            // ou en mode 1600pas/rev : 8 pulses pour un pas avec un␣
↪motor de 1.8deg
                            // ou 200pas/rev)
#define dir_pin 8           // Pin 8 connected to Direction pin
#define home_switch 12      // Pin 12 connected to Home Switch (MicroSwitch)
#define ref 0               // c'est le 0 ici l'inversion est faite en mode basse␣
↪vitesse
                            // (sans accélération ni décélération)

//
↪**********************************************************************************************
//*********************PARAMETRES␣
↪MODIFIABLES*****************************************************
#define v_max 40            // C'est la vitesse maximale lue par le potentiomètre de␣
↪vitesse (délai en
                            // microseconde entre deux pulses) 30 50 45 55 65 70 65␣
↪Valeur original 50.
#define v_min 1200          // C'est la vitesse minimum lue par le potentiomètre de␣
↪vitesse (délai en
                            // microseconde entre deux pulses)
#define v_max_inv 200       // C'est la vitesse maximale à laquelle le moteur ne␣
↪décroche pas 200 280
                            // Valeur précédente=200 à l'inversion de sens de␣
↪rotation (200 pour Gecko
                            // et 200 pour 1600pas/rev?). En dessous de cette vitesse,
↪ l'accélération
                            // et la décélération ne sont pas requise. C'est pourquoi␣
↪la course peut être
                            // plus petite car il n'y pas besoin de distance de␣
↪freinage. A cette petite
                            // course le va-et-vient est tellement rapide que ça␣
↪vibre. On retrouve ici
                            // le mode vibro.
#define course_max 2968     // C'est la course maximale lue par le potentiomètre de␣
↪course 2150 2125
                            // 2100 2080 origine 2320 2500 Cette valeur dépende de la␣
↪longueur de la
                            // machine.

#define marge 30            // C'est la marge ou décalage de la référence par rapport␣
↪à la position du
```

(continues on next page)

(continued from previous page)

```
                              // bras à la mise sous tension de la machine en principe␣
↪en butée du coté
                              // vert ou coté rouge. Ceci assure que la partie mobile␣
↪ne viennent cogné
                              // sur l'un des cotés. Valeur originale = 50.
//
↪*****************************************************************************************
int course_min = 296;         // C'est la course minimum lue par le potentiomètre de␣
↪course. Qui corresond
                              // à la course minimum pour l'accélération et la␣
↪décélération à haute vitesse.
                              // (vitesse à laquelle la distance de freinage/
↪accélération est requise).
                              // Soit course_min > 2*(v_max_inv - v_max) = distance max␣
↪de freinage ou
                              // d'accélération.

                             // Si on est dans le mode basse vitesse alors on peut␣
↪réduire la course pour
                              // le mode vibro.
int diff_course = 1200;      // C'est la tolérance d'une différence de course. Cette␣
↪valeur est utilisée pour
                              // comparer la valeur précédente de la course␣
↪(customCourseMapped_pre) et la
                              // valeur courante. S'il y avait un problème de mesure du␣
↪poto cela évite que
                              // la course se mette au max, ce qui pourrait être␣
↪dangereux.
int x=v_max_inv;              // Variable de délai entre impulsion qui varie selon␣
↪accéleration et décélération
int steps=0;                  // Variable compteur de pas. Sert à savoir la position du␣
↪bras (position courante)
int steps_init=0;             // Variable compteur de pas. Pour la prise de référence à l
↪'initialisation

int analog_read_counter = 300; // Permet de pas lire trop souvent la vitesse du␣
↪potentiometre. Ce qui permet
                               // une vitesse plus élevée du va-et-vient.

int customCourseMapped;        // Pour stocker la consigne de la course courante
int customCourseMapped_pre;    // Pour stocker la consigne de la course précédente␣
↪(pour controler que la diff
                               // n'est pas trop grande (sécurité)
int customSpeedMapped;         // Pour stocker la consigne de la vitesse
int dist_frein = v_max_inv-v_max; // C'est la vitesse de freinage. On initialise à la␣
↪valeur max (quand la vitesse est la plus grande)

bool CoteVert = false;   // Pour définir le côté vert avec les potos
bool CoteRouge = false;  // Pour définir le côté rouge avec les potos
bool CoteMilieu = false; // Pour définir l'utilisation à deux de la machine (les bras␣
↪se mettent au milieux.

void setup() {
  pinMode(dir_pin, OUTPUT);
  pinMode(step_pin, OUTPUT);
  digitalWrite(dir_pin, LOW);
  digitalWrite(step_pin, LOW);
```

(continues on next page)

(continued from previous page)

```
  //Serial.begin(9600);

  pinMode(home_switch, INPUT_PULLUP);

  delay(1000);  // on attend un peu

  steps=-marge;  // on rajoute un poil de cul plié en quatre pour que le 0 soit un␣
↪peu plus loin
                 // on aura, avant la mise sous tension de la machine, poussé le bras␣
↪en butée à
                 // l'intérieur de la machine du côté vert ou rouge (côté duquel on␣
↪veut se mettre)

  //
↪**********************************************************************************
  // ici on vérifie la position des potos afin de savoir de quel côté la machine est␣
↪utilisée.
  //
↪**********************************************************************************

  //ici la machine est utilisée du côté vert

  if(speedUp() > 1100 && course() < 400){
  CoteVert = true;
  CoteRouge = false;}

  //ici la machine est utilisée du côté rouge

  if(speedUp() < 100 && course() > 2600){
    CoteRouge = true;
    CoteVert = false;
  }

  //ici la machine est utilisée sur les deux côtés à la fois

  if((speedUp() > 400 && speedUp() < 800) && (course() > 1290 && course() < 2090)){
    CoteMilieu = true;
    CoteVert = false;
    CoteRouge = false;
  }

  if(CoteMilieu){
    while(steps<(course_max/2)){  // va jusqu'au milieu
      digitalWrite(step_pin, HIGH);
      delay(1);
      digitalWrite(step_pin, LOW);
      delay(1);
      steps++;
    }
  }

  // tant que le potentiomètre de la vitesse et de la course ne sont pas au minimum,␣
↪on ne démarre pas (pour la sécurité)

  while (!(speedUp() > (v_min - 30) && course() < 400)){
  }
```

(continues on next page)

(continued from previous page)

```
}

void loop() {

  customSpeedMapped =speedUp(); // lecture de la vitesse
  customCourseMapped = course(); // lecture de la course
  if (((customCourseMapped-customCourseMapped_pre) > diff_course) &&␣
↪(customSpeedMapped < 200)) {
    while(1){} // On rentre en boucle infinie (pour la sécurité évite de se faire␣
↪percer trop violemment)
  }


  //*****************************************Ne sait pas trop ce que ça fait?
↪*************************
  while ((customSpeedMapped) > (v_min - 17)){ // si le potentiomètre de la vitesse␣
↪est au minimum,
                                              //on met LM42P à l'arrêt
    if (analog_read_counter > 0) { //pour ne pas lire trop souvent le pot de la␣
↪vitesse
      analog_read_counter--;
    }
    else {
      analog_read_counter = 400;
      customSpeedMapped =speedUp(); // permet de modifier la vitesse sans devoir␣
↪attendre un va complet
    }
  }
 //
↪*****************************************************************************************************


 if(CoteVert){
  if(customSpeedMapped<v_max_inv){
    dist_frein = v_max_inv-customSpeedMapped;                                  // On␣
↪recalcule la distance de freinage pour la vitesse de consigne actuelle.
    course_min = 380;      //course min plus grande                            // Plus␣
↪la vitesse est élévée et plus la distance de freinage est longue.
  }                                                                            // Rappel␣
↪: une valeur petite pour la vitesse signifie une grande vitesse
                                                                               //␣
↪puisque la valeur est le délai en microsecondes entre-pulse                   ␣
↪
  else{course_min = 30;} // fonction vibro à basse vitesse

 // customCourseMapped = course(); // on lit la course en fonction vibro ou␣
↪pas***********************

  digitalWrite(dir_pin, LOW);
  if (customSpeedMapped>200) { // Mode basse vitesse. Ici l'accélération et la␣
↪décélération ne sont pas requise.

     while(steps<customCourseMapped){  // va jusqu'à course
       digitalWrite(step_pin, HIGH);
       delayMicroseconds(customSpeedMapped);
       digitalWrite(step_pin, LOW);
       delayMicroseconds(customSpeedMapped);
```

(continues on next page)

(continued from previous page)

```
      steps++;
      //Serial.println(steps);
      // We only want to read the pot every so often (because it takes a long time
↪we don't
      // want to do it every time through the main loop).
      if (analog_read_counter > 0) {
        analog_read_counter--;
      }
      else {
        analog_read_counter = 300;
        customSpeedMapped =speedUp(); // permets de modifier la vitesse sans devoir
↪attendre un va complet
      }
    }

    digitalWrite(dir_pin, HIGH); //changement de direction et va jusqu'à la
↪référence 0
    while(steps>ref){
      digitalWrite(step_pin, HIGH);
      delayMicroseconds(customSpeedMapped);
      digitalWrite(step_pin, LOW);
      delayMicroseconds(customSpeedMapped);
      steps--;
      //Serial.println(steps);
      // We only want to read the pot every so often (because it takes a long time
↪we don't
      // want to do it every time through the main loop).
      if (analog_read_counter > 0) {
        analog_read_counter--;
      }
      else {
        analog_read_counter = 300;
        customSpeedMapped =speedUp(); // permets de modifier la vitesse sans devoir
↪attendre un va complet
      }
    }
  }

  else{

  // ici la vitesse rapide est activée et l'accélération et la décélértion est requise

  digitalWrite(dir_pin, LOW);

  //accélération et va jusqu'à course
  while(steps<(customCourseMapped-dist_frein)){ // On doit réduire la course pour
↪laisser de la distance au freinage
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x>customSpeedMapped) x--;  // tant que x est plus grand (plus x est grand plus
↪la vitesse est lente) que la vitesse du poto, on accélère
    steps++;
  }
  // Là on a atteint la course - la distance de freinage
```

(continues on next page)

(continued from previous page)

```
  //décélération
  while (x<v_max_inv){
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x<v_max_inv) x++;
     steps++; //on doit continuer à incrémenter car ça continue à avancer
  }

  digitalWrite(dir_pin, HIGH); //Changement de direction
  while(steps>(ref+dist_frein)){ // Acccélération et va jusqu'à la référence +␣
↪distance de freinage.
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x>customSpeedMapped) x--;
    steps--;
  }

  while(x<(v_max_inv)){ //décélération
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x<v_max_inv) x++;
    steps--;
  }
//*/
  }
  //steps=steps+2; //pour compenser je ne sais pas quoi? pour le petit driver il faut␣
↪mettre +2 et rien pour le grand (commenter)
}

 if(CoteRouge){
  if(customSpeedMapped<v_max_inv){
    dist_frein = v_max_inv-customSpeedMapped;                                  // On␣
↪recalcule la distance de freinage pour la vitesse de consigne actuelle.
    course_min = 380;      //course min plus grande                            // Plus␣
↪la vitesse est élévée et plus la distance de freinage est longue.
  }                                                                            // Rappel␣
↪: une valeur petite pour la vitesse signifie une grande vitesse
                                                                               //␣
↪puisque la valeur est le délai en microsecondes entre-pulse                   ␣
↪
  else{course_min = 30;} // fonction vibro à basse vitesse

  //customCourseMapped = course(); // on lit la course en fonction vibro ou␣
↪pas*****************************

  digitalWrite(dir_pin, HIGH);
  if (customSpeedMapped>200) { // Mode basse vitesse. Ici l'accélération et la␣
↪décélération ne sont pas requise.
```

(continues on next page)

(continued from previous page)

```
    while(steps<customCourseMapped){  // va jusqu'à course
     digitalWrite(step_pin, HIGH);
     delayMicroseconds(customSpeedMapped);
     digitalWrite(step_pin, LOW);
     delayMicroseconds(customSpeedMapped);
     steps++;
     //Serial.println(steps);
     // We only want to read the pot every so often (because it takes a long time we␣
↪don't
     // want to do it every time through the main loop).
     if (analog_read_counter > 0) {
       analog_read_counter--;
     }
     else {
      analog_read_counter = 300;
      customSpeedMapped =speedUp(); // permets de modifier la vitesse sans devoir␣
↪attendre un va complet
    }
  }
  //*
  digitalWrite(dir_pin, LOW); //changement de direction et va jusqu'à la référence 0
  while(steps>ref){
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(customSpeedMapped);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(customSpeedMapped);
    steps--;
    //Serial.println(steps);
    // We only want to read the pot every so often (because it takes a long time we␣
↪don't
    // want to do it every time through the main loop).
    if (analog_read_counter > 0) {
      analog_read_counter--;
    }
    else {
      analog_read_counter = 300;
      customSpeedMapped =speedUp(); // permets de modifier la vitesse sans devoir␣
↪attendre un va complet
    }
  }
}

  else{

    // ici la vitesse rapide activée et l'accélération et la décélértion est requise

  digitalWrite(dir_pin, HIGH);

  //accélération et va jusqu'à course
  while(steps<(customCourseMapped-dist_frein)){ // On doit réduire la course pour␣
↪laisser de la distance au freinage
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
```

(continues on next page)

(continued from previous page)

```
    if(x>customSpeedMapped) x--;  // tant que x est plus grand (plus x est grand plus␣
↪la vitesse est lente) que la vitesse du poto, on accélère
    steps++;
  }
  // Là on a atteint la course - la distance de freinage
  //décélération
  while (x<v_max_inv){
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x<v_max_inv) x++;
     steps++; //on doit continuer à incrémenter car ça continue à avancer
  }

  digitalWrite(dir_pin, LOW); //Changement de direction
  while(steps>(ref+dist_frein)){ // Acccélération et va jusqu'à la référence +␣
↪distance de freinage.
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x>customSpeedMapped) x--;
    steps--;
  }

  while(x<(v_max_inv)){ //décélération
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x<v_max_inv) x++;
    steps--;
  }
//*/
  }
  //steps=steps+2; //pour compenser je ne sais pas quoi? pour le petit driver il faut␣
↪mettre +2 et rien pour le grand (commenter)
}
if(CoteMilieu){
  if(customSpeedMapped<v_max_inv){
    dist_frein = v_max_inv-customSpeedMapped;                                  // On␣
↪recalcule la distance de freinage pour la vitesse de consigne actuelle.
    course_min = 380;      //course min plus grande                            // Plus␣
↪la vitesse est élévée et plus la distance de freinage est longue.
  }                                                                            // Rappel␣
↪: une valeur petite pour la vitesse signifie une grande vitesse
                                                                               //␣
↪puisque la valeur est le délai en microsecondes entre-pulse                      ␣
↪
  else{course_min = 30;} // fonction vibro à basse vitesse

 //customCourseMapped = course(); // on lit la course en fonction vibro ou␣
↪pas**********************
```

(continues on next page)

(continued from previous page)

```
  digitalWrite(dir_pin, LOW);
  if (customSpeedMapped>200) { // Mode basse vitesse. Ici l'accélération et la␣
↪décélération ne sont pas requise.

     while(steps<((course_max/2)+(customCourseMapped)/2)){  // va jusqu'à course
      digitalWrite(step_pin, HIGH);
      delayMicroseconds(customSpeedMapped);
      digitalWrite(step_pin, LOW);
      delayMicroseconds(customSpeedMapped);
      steps++;
      //Serial.println(steps);
      // We only want to read the pot every so often (because it takes a long time we␣
↪don't
      // want to do it every time through the main loop).
      if (analog_read_counter > 0) {
        analog_read_counter--;
      }
      else {
       analog_read_counter = 300;
       customSpeedMapped =speedUp(); // permets de modifier la vitesse sans devoir␣
↪attendre un va complet
     }
  }
  //*
  digitalWrite(dir_pin, HIGH); //changement de direction et va jusqu'à la référence 0
  while(steps>((course_max/2)-(customCourseMapped/2))){
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(customSpeedMapped);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(customSpeedMapped);
    steps--;
    //Serial.println(steps);
    // We only want to read the pot every so often (because it takes a long time we␣
↪don't
    // want to do it every time through the main loop).
    if (analog_read_counter > 0) {
      analog_read_counter--;
    }
    else {
      analog_read_counter = 300;
      customSpeedMapped =speedUp(); // permets de modifier la vitesse sans devoir␣
↪attendre un va complet
    }
  }
}

  else{

    // ici la vitesse rapide activée et l'accélération et la décélértion est requise

  digitalWrite(dir_pin, LOW);

  //accélération et va jusqu'à course
  while(steps<((course_max/2)+(customCourseMapped/2)-dist_frein)){ // On doit réduire␣
↪la course pour laisser de la distance au freinage steps<((course_max/
↪2)+(customCourseMapped/2)-dist_frein)
```

(continues on next page)

(continued from previous page)

```
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x>customSpeedMapped) x--;  // tant que x est plus grand (plus x est grand plus la vitesse est lente) que la vitesse du poto, on accélère
    steps++;
  }
  // Là on a atteint la course - la distance de freinage
  //décélération
  while (x<v_max_inv){
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x<v_max_inv) x++;
     steps++; //on doit continuer à incrémenter car ça continue à avancer
  }

  digitalWrite(dir_pin, HIGH); //Changement de direction
  while(steps>((course_max/2)-(customCourseMapped/2)+dist_frein)){ // Acccélération et va jusqu'à la référence + distance de freinage. steps>((course_max/2)-(customCourseMapped/2))
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x>customSpeedMapped) x--;
    steps--;
  }

  while(x<(v_max_inv)){ //décélération
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(x);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(x);
    //Serial.println(steps);
    if(x<v_max_inv) x++;
    steps--;
  }
//*/
  }
  //steps=steps+2; //pour compenser je ne sais pas quoi? pour le petit driver il faut mettre +2 et rien pour le grand (commenter)
}
customCourseMapped_pre = customCourseMapped; //pour la calcul de la diff (sécurité)
}
  // Function for reading the Potentiometer
    int speedUp() {
      int customSpeed = analogRead(A0); // Reads the potentiometer
      int newCustom = map(customSpeed, 1023, 0, v_max,v_min);
      return newCustom;
   }
```

(continues on next page)

(continued from previous page)

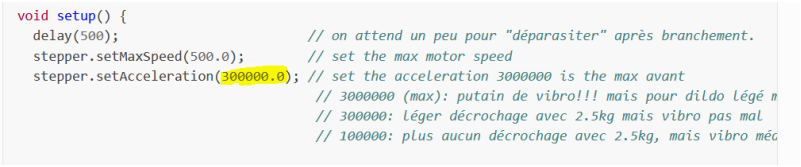
```
    int course() {
      int customCourse = analogRead(A1); // Reads the potentiometer
      int newCustom2 = map(customCourse, 1023, 0, course_max,course_min);
      return newCustom2;
    }
```

3.2 CodeGeckoAccelStepVx.ino

Note:

For a better vibro you can adjust that parameter **stepper.setAcceleration(....)**:

- 3000000 is the fastest acceleration for a strong vibro but for light dildo ;
- 300000 for up to 2kg dildo ;
- 100000 for heavier dildos.




```
// CodeGeckoAccelStepV1.ino
// V1: Version de base
// Auteur: LM42P
// Date: 4 july 2022
// Driver used: Geckodrive G201X
// Motor: Stepper Motor NEMA 23 23HS11240
// Arduino: Arduino Uno
//
// Ce programme utilise la librairie AccelStepper pour un changement de sens plus␣
↪doux. Il supprime
// le saut de vitesse par rapport au CodeGeckoVx.ino, au détriment du vitesse plus␣
↪faible.

#include <AccelStepper.h>           // Load the AccelStepper library

// direction Digital 8 (CCW), pulses Digital 9 (CLK)
AccelStepper stepper(1, 9, 8);

  //
↪**********************************************************************************************************
  //**********************PARAMETRES␣
↪MODIFIABLES*************************************************

  #define v_max 5000          // C'est la vitesse maximale lue par le potentiomètre de␣
↪vitesse. Peut être
                              // descendue à 4800, car il y un bout vers la fin qui ne␣
↪va pas plus vite.
  #define v_min 120           // C'est la vitesse minimum lue par le potentiomètre de␣
↪vitesse.
```

(continues on next page)

(continued from previous page)

```
  #define course_max_physique 3044 // C'est la course maximale physique de la machine␣
↪(butée à butée
                           // sans jeu). Cette valeur a été détérminée par la␣
↪programme setFullStroke.
  #define course_max course_max_physique-2*marge // C'est la valeur de la course␣
↪effective (tenant
                           // compte des marges. Cette valeur sert à mapper le␣
↪potentiomètre
                           // de course.

//       |<-----------------------------------course_max_physique--------------------
↪----------->|
//       |<--marge-->|<-----------------------------course_max----------------------->
↪|<--marge-->|
//                   |<-->|                                                      |<-->
↪|
//                 course_min                                                  course_
↪min

//       |<--marge-->|<---------course_max/2--------->|<---------course_max/2-------->
↪|<--marge-->|

//                                                |<->|<->|
//                                           course_min/2 course_min/2

  #define course_min 40     // c'est la course minimum pour que le vibro soit␣
↪efficace. Une valeur trop
                           // petite donnera des amplitudes de variation trop␣
↪petites.
  #define marge 15          // C'est la marge ou décalage de la référence par rapport␣
↪à la position du
                           // bras à la mise sous tension de la machine en butée␣
↪soit du coté vert ou
                           // rouge. Ceci assure que la partie mobile ne viennent␣
↪cogné sur l'un des
                           // cotés. J'ai baissé cette valeur de 30 à 15, pour␣
↪profiter de la course
                           // au max.


 // Points définissant les plages sur les potentiomètres (voir schéma):

  #define i 0.2 // facteur de dimensionnement de la plage
  #define A_course course_min // on met une marge de 10 pour être sûr que l'on soit␣
↪dans la plage
                              // en butée
  #define B_course A_course+plage_course
  #define C_course (F_course-A_course)/2+A_course-plage_course/2
  #define D_course (F_course-A_course)/2+A_course+plage_course/2
  #define E_course F_course-plage_course
  #define F_course course_max // on met une marge de 10 pour être sûr que l'on soit␣
↪dans la plage
                              // en butée
  #define plage_course i*(F_course-A_course)

  bool plage_min_course=false;
  bool plage_milieu_course=false;
```

(continues on next page)

(continued from previous page)

```
  bool plage_max_course=false;

  #define A_vitesse v_min // on met une marge de 10 pour être sûr que l'on soit dans la plage
                            // en butée
  #define B_vitesse A_vitesse+plage_vitesse
  #define C_vitesse (F_vitesse-A_vitesse)/2+A_vitesse-plage_vitesse/2
  #define D_vitesse (F_vitesse-A_vitesse)/2+A_vitesse+plage_vitesse/2
  #define E_vitesse F_vitesse-plage_vitesse
  #define F_vitesse v_max // on met une marge de 10 pour être sûr que l'on soit dans la plage
                            // en butée
  #define plage_vitesse i*(F_vitesse-A_vitesse)

  bool plage_min_vitesse=false;
  bool plage_milieu_vitesse=false;
  bool plage_max_vitesse=false;

  bool CoteVert = false;   // Pour définir le côté vert avec les potos
  bool CoteRouge = false;  // Pour définir le côté rouge avec les potos
  bool Milieu = false;     // Pour définir l'utilisation à deux de la machine
                           // (les bras se mettent au milieu)

  int cours=0; // C'est la course
  int customSpeedMapped = 5;          // Pour stocker la consigne de la vitesse
  int analog_read_counter = 300;      // Compteur est utilisé pour ne de pas lire la vitesse du potentiometre
                                      // trop souvent. Ce qui permet une vitesse plus élevée du va-et-vient
                                      // et évite les parasites.
                                                                                  

void setup() {
  delay(500);                         // on attend un peu pour "déparasiter" après branchement.
  stepper.setMaxSpeed(500.0);         // set the max motor speed
  stepper.setAcceleration(300000.0); // set the acceleration 3000000 is the max avant
                                       // 3000000 (max): putain de vibro!!! mais pour dildo légé max 800g
                                       // 300000: léger décrochage avec 2.5kg mais vibro pas mal
                                       // 100000: plus aucun décrochage avec 2.5kg, mais vibro médiocre

//
//////////////////////////////////////////////////////////////////////////////////////////
  // Ici on vérifie la position des potensiomètres pour savoir de quel côté la machine
  // est utilisée. (voir schéma)
  //
//////////////////////////////////////////////////////////////////////////////////////////

//Le curseur de potentiomètre de la course est dans la plage min
  if (course()>=A_course-10 && course()<B_course){
    plage_min_course=true;
    plage_milieu_course=false;
    plage_max_course=false;
  }
```

(continues on next page)

(continued from previous page)

```
//Le curseur de potentiomètre de la course est dans la plage milieu
  else if (course()>C_course && course()<D_course){
    plage_min_course=false;
    plage_milieu_course=true;
    plage_max_course=false;
  }

//Le curseur de potentiomètre de la course est dans la plage max
  else if (course()>E_course && course()<=F_course+10){ //+10 plage plus grande
    plage_min_course=false;
    plage_milieu_course=false;
    plage_max_course=true;
  }
  else {
    plage_min_course=false;
    plage_milieu_course=false;
    plage_max_course=false;
  }

//Le curseur de potentiomètre de la vitesse est dans la plage min
  if (speedUp()>=A_vitesse-10 && speedUp()<B_vitesse){ //-10 plage plus grande
    plage_min_vitesse=true;
    plage_milieu_vitesse=false;
    plage_max_vitesse=false;
  }

//Le curseur de potentiomètre de la vitesse est dans la plage milieu
  else if (speedUp()>C_vitesse && speedUp()<D_vitesse){
    plage_min_vitesse=false;
    plage_milieu_vitesse=true;
    plage_max_vitesse=false;
  }

//Le curseur de potentiomètre de la vitesse est dans la plage max
  else if (speedUp()>E_vitesse && speedUp()<=F_vitesse+10){ //+10 plage plus grande
    plage_min_vitesse=false;
    plage_milieu_vitesse=false;
    plage_max_vitesse=true;
  }
  else {
    plage_min_vitesse=false;
    plage_milieu_vitesse=false;
    plage_max_vitesse=false;
    }

  //ici la machine est utilisée du côté vert

  if(plage_min_course && plage_min_vitesse){
    CoteVert = true;
    Milieu = false;
    CoteRouge = false;
  }

  //ici la machine est utilisée sur les deux côtés à la fois

  else if(plage_milieu_course && plage_milieu_vitesse){
```

(continues on next page)

(continued from previous page)

```
    CoteVert = false;
    Milieu = true;
    CoteRouge = false;
  }


  //ici la machine est utilisée du côté rouge

  else if(plage_max_course && plage_max_vitesse){
    CoteVert = false;
    Milieu = false;
    CoteRouge = true;
  }

  else{
    CoteVert = false;
    Milieu = false;
    CoteRouge = false;
    }

// on met les bonnes références suivant le côté voulu

  //ici la machine est utilisée du côté vert
  if(CoteVert){

    // On référence la position à marge (pour laisser de la place à la butée)
    stepper.setCurrentPosition(marge);

    // On se déplace à 0.
    stepper.moveTo(0);
    while (!stepper.distanceToGo() == 0) {
    stepper.run();
    }
  }

//ici la machine est utilisée sur les deux côtés à la fois (potentiomètres au milieu)

  else if(Milieu){
    // On référence la position à marge (pour laisser de la place à la butée)
    stepper.setCurrentPosition(course_max_physique/2);

    // On se déplace à 0
    stepper.moveTo(0);
    while (!stepper.distanceToGo() == 0) {
    stepper.run();
    }
  }

//ici la machine est utilisée du côté rouge
  else if(CoteRouge){
    //On reset la position à -marge (pour laisser de la place à la butée)
    stepper.setCurrentPosition(-marge);

    // On se déplace à 0.
    stepper.moveTo(0);
    while (!stepper.distanceToGo() == 0) {
      stepper.run();
```

(continues on next page)

(continued from previous page)

```
    }
  }

  // Tant que le potentiomètre de la vitesse et de la course ne sont pas au minimum, on ne
  // démarre pas (pour la sécurité)
  while (!(speedUp() < (v_min + 30) && course() < course_min+100)){}

  } //fin du setup

void loop() {

//*********si le potentiomètre de la vitesse est au minimum, on met LM42P à l'arrêt**************
  while ((customSpeedMapped) < (v_min +7)){

    if (analog_read_counter > 0) { //pour ne pas lire trop souvent le pot de la vitesse
      analog_read_counter--;
    }
    else {
      analog_read_counter = 300;
      customSpeedMapped =speedUp();
    }
  } // fin de la boucle while //mettre la machine à l'arrêt

  // on règle la vitesse de la machine
  if (analog_read_counter > 0) { //pour ne pas lire le pot de la vitesse trop souvent et évite les parasites
      analog_read_counter--;
  }
  else {
    analog_read_counter = 300;
    customSpeedMapped =speedUp();
    stepper.setMaxSpeed(customSpeedMapped); // permet de modifier la vitesse sans devoir attendre
                                            // un va-et-vient complet
  }  // fin on règle la vitesse de la machine

  if(CoteVert){ // si la machine est utilisée du côté vert
  // If the stepper isn't moving and doesn't have to go any distance
    if (!stepper.isRunning() && stepper.distanceToGo() == 0) {
      stepper.moveTo(-cours);
      if (cours>0){
        cours=0;
      }
      else cours=course();
    }
  }

  else if(Milieu){ // si la machine est utilisée des deux côtés à la fois
  // If the stepper isn't moving and doesn't have to go any distance
    if (!stepper.isRunning() && stepper.distanceToGo() == 0) {
      stepper.moveTo(cours);
      cours=(-1*cours);
      if (cours>=0){
        cours=course()/2;
```

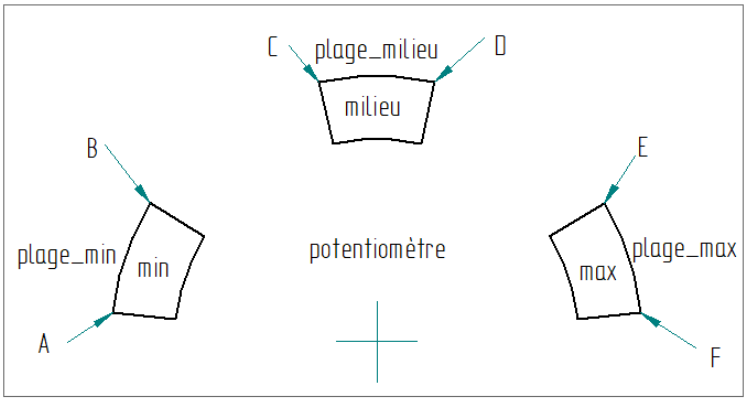
(continues on next page)

(continued from previous page)

```
      }
    }
  }

  else if(CoteRouge){ // si la machine est utilisée du côté rouge
  // If the stepper isn't moving and doesn't have to go any distance
    if (!stepper.isRunning() && stepper.distanceToGo() == 0) {
      stepper.moveTo(cours);
      if (cours>0){
        cours=0;
      }
      else cours=course();
    }
  }

  stepper.run();

} //fin de la boucle principale

  // Function for reading the Potentiometer
    int speedUp() {
      int customSpeed = analogRead(A0); // Reads the potentiometer
      int newCustom = map(customSpeed, 0, 1023, v_min,v_max);
      return newCustom;
   }

    int course() {
      int customCourse = analogRead(A1); // Reads the potentiometer
      int newCustom2 = map(customCourse, 0, 1023, course_min, course_max);
      return newCustom2;
    }
```

3.2.1 Figure

This figure helps to understand how the area on remote are defined.



A = course_min ou speed_min
F = course_max ou speed_max
B = A + plage
C = (F – A) / 2 + A – plage / 2
D = (F – A) / 2 + A + plage / 2
E = F – plage
plage = i x (F – A)
i = 20%

3.3 Updates

V3 -> V4 : In the V3 when the RJ45 (Remote Control) is pull out during full speed, the stroke goes to its max, this could be dangerous. V4 avoid this and a reset is necessarily when it happens. Also V4 doesn't permit a too much high gradient of stroke change.

V4->V5 : V4->V5 stroke_min has been set to 296 instead of 380 allows maybe to have a better vibro

V1: first version

3.4 How to update the Latest Firmware

This video helps to understand how to update Firmware :

1. Download and install Arduino IDE software
2. Connect LM42P with a usb cable
3. Open Arduino IDE
4. File New
5. Delete what is in the file
6. Copy / paste the latest code from LM42P site
7. Click compile button there should be no error
8. Click upload button

Note: If there is a error during uploading then you should try to select the good Port. Go to Tools -> Ports If Ports is grey then try to put another arduino the port should not be grey. Then put the arduino you wanted to update.

CHAPTER 4

Accessories

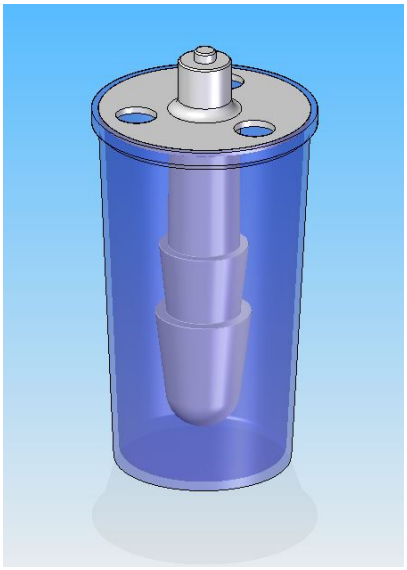
4.1 Suction Cup Dildo Adapter

This is the files to print the Suction Cup Dildos Adapter. It is made of two parts: one is the above and the under part. With a screw it is clamped on a aluminium scare tube of size 15mm.

STL file ->

4.2 Vac u Lock Adapter

4.2.1 Silicon Mold



Listing Parts

1) Cup
2) Centerign Cover
3) Threaded Stem
4) Aluminium Vac u Lock Model

Operation Plan

1) Prepare 220g silicon and 11g hardener
2) Stir 5min
3) Put a tape border on the plastic cup
4) Wait 20min to remove the bubles
5) Fill in through one of the hole

4.2.2 Mold the Adapter

1) Use a 50mm long M6 bolt
2) Prepare 20g cast and 20g hardener
3) Stir really slowly

CHAPTER 5

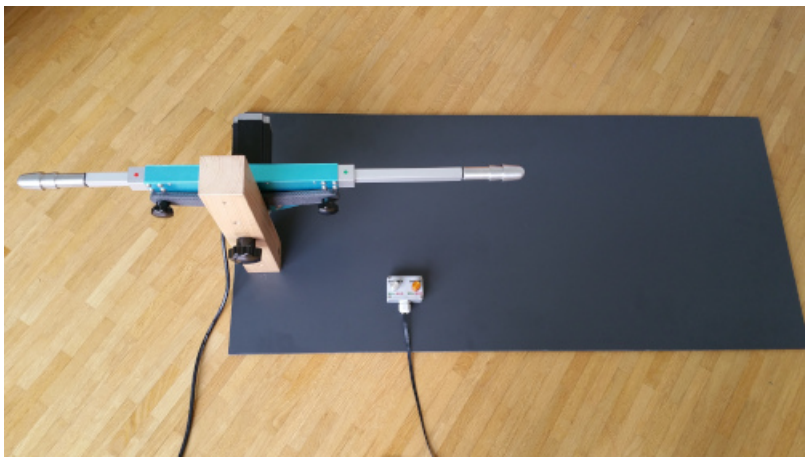
Fastening Elements

5.1 Video

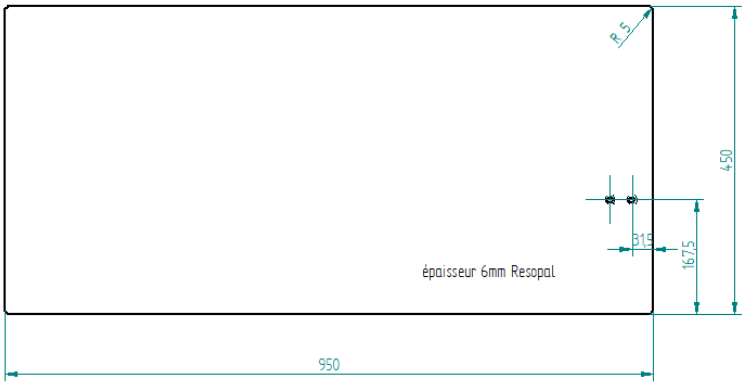
5.2 Mast

5.2.1 Tools

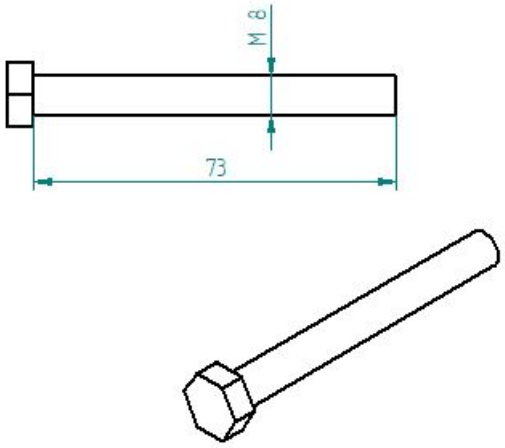
5.3 Building Plate



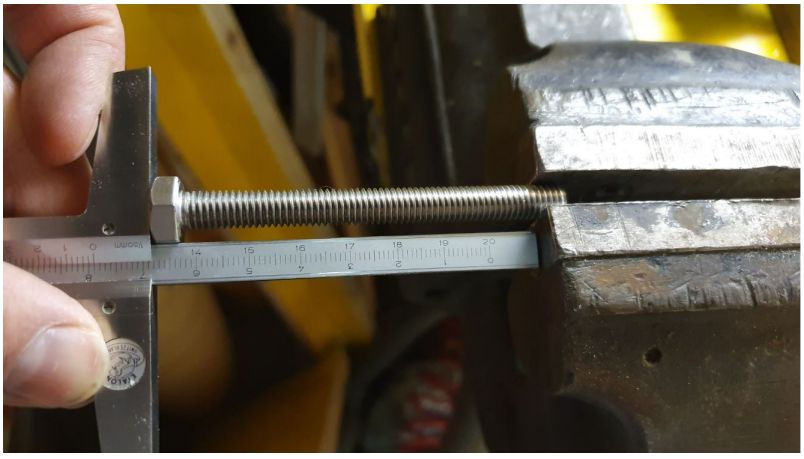
- Material : RESOPAL
- Thinkness : 6mm
- Size : 450 x 950 x 6 mm
- Weight : 4kg



5.4 Fastening bolt

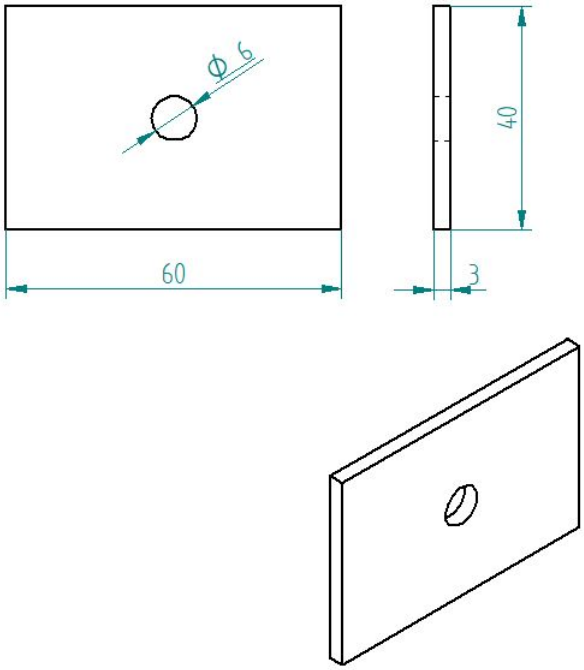


1. Saw the bolt



2. Deburr the bolt

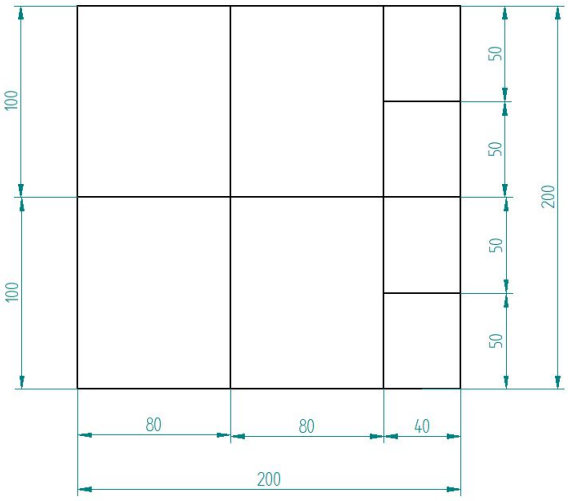
5.5 Silicon-Protection 60x40x3



1. Cut the plate with a cutter
2. Cut wiht a cookie cutter the hole diameter 6mm

5.6 Silicon-Protection 80x100x8

From a silicon plate 200x200x8 cut 4x parts 80x100 see following figure.



5.7 Silicon-Protection 40x50x8

From a silicon plate 200x200x8 cut 4x parts 40x50 mm see preceding figure.

5.8 Allen Cone Head Screw

- Type : M8 x 45mm
- Supplier : Debrunner
- N° Article : 101 14 091

5.9 Allen Tightening Key

- Type : 5mm
- Supplier : Debrunner
- N° Article : 811 305 240

CHAPTER 6

Storage Box

6.1 Video

See the video that shows how to make the Storage Box step by step.

6.2 Listing Parts

- Wooden Screw 2.5/9
  - quantity = 6x
  - diameter = 2.5mm
  - length = 9mm
  - head diameter = 5mm

- Wooden Screw 2/17
  - quantity = 6x
  - diameter = 2.5mm

  - length = 17mm
  - head diameter = 4mm

- Wooden Screw Hole M4
  - quantity = 4x
  - diameter = M4

- Screw M4
  - quantity = 4x
  - diameter = M4
  - length = 10mm

- Hinge
  - quantity = 1x

- Handle
  - quantity = 2x
  - material = stainless

- Womdee 3mm
  - quantity = a lot
  - material = wood

- Double-sided Tape

- Wood Glue

CHAPTER 7

Donate

Please give me a small tip to help me continue this self-buiding site! Click the following Paypal.me Icon or subscribe to my Youtube Channel.

Thank You very much!

CHAPTER 8

Contact

Address :

LM42P

1010 Lausanne Switzerland

internet :

https://lm42p.com

email :

lovemachine42people@gmail.com

CHAPTER 9

Indices and tables

- genindex
- modindex
- search