



Tutorial for creating housings in FreeCAD

☆ 0 stars 🍴 0 forks 👁 1 watching ↻ Activity

🌐 Public repository

🔗 **main** ▾



🔗 Branches 🏷 Tags



henkjannl Readme ...

7 minutes ago ⌚ 12

[View code](#)

Introduction

This tutorial demonstrates how to design 3D printable housings in FreeCAD. The advantage of this approach is that it is quite structured, and you can manage changes in the design quite well, even if the design gets complex.

To follow this tutorial, you need to be familiar with the part design workbench and the sketcher. I'll try to just focus on a high conceptual level.

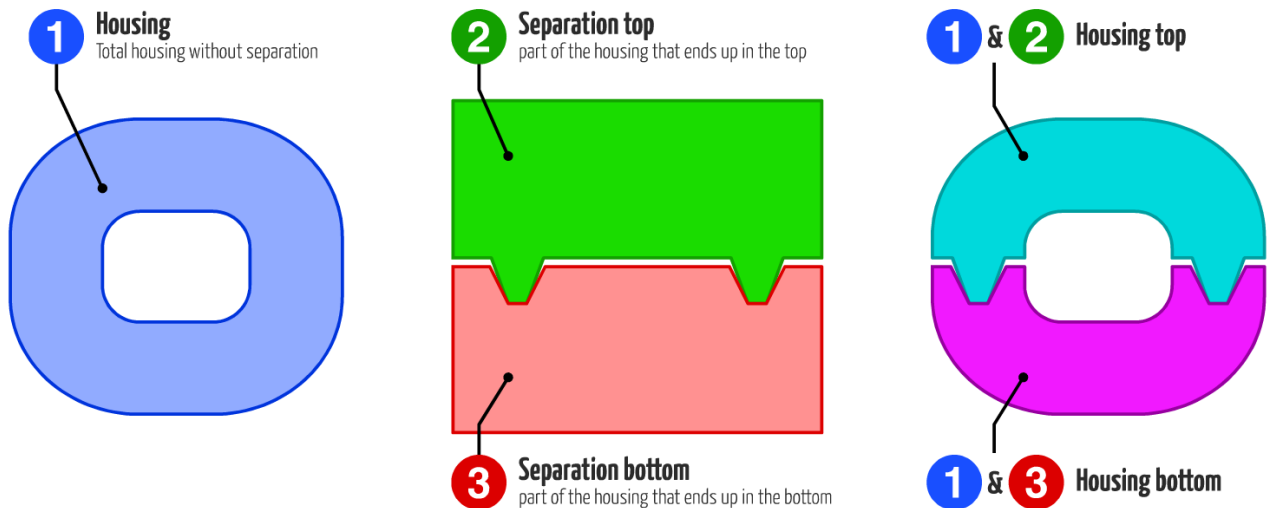
Topics in this document

- [Introduction](#)
- [Topics in this document](#)
- [Concept of making a housing using boolean operation of bodies](#)
- [Making changes to the housing](#)
- [Maintaining the colors of both housing bodies](#)
- [Applying a naming convention for bodies and features](#)
- [Using a skeleton to drive dimensions of the bodies](#)
- [Checking the model](#)
 - [Checking links](#)
 - [Using the Check geometry tool](#)
 - [Checking the result in the slicer](#)
- [Creating references to the internal components of the housing](#)
- [Using self tapping screws to close the housing](#)
 - [Creating a screw hole](#)
 - [Creating a pillar for the screw](#)
- [Creating a complex hinge](#)
- [Creating references to external parts](#)

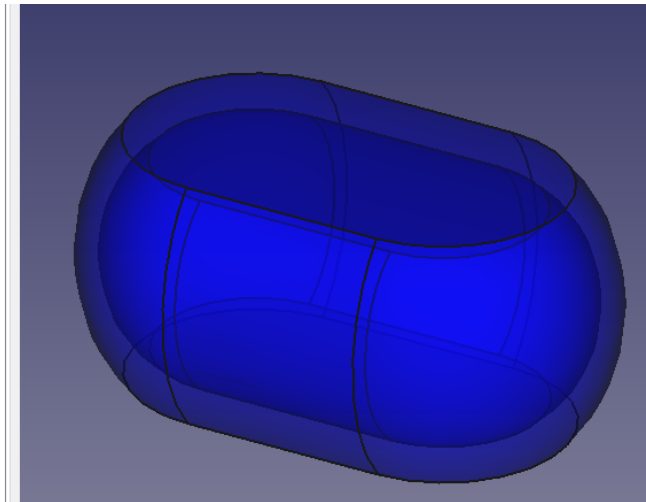
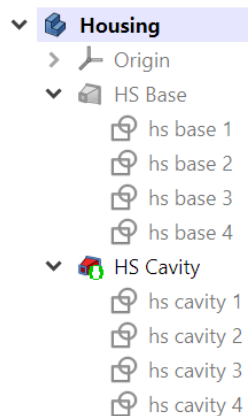
Concept of making a housing using boolean operation of bodies

In this example, we want to create a housing which consists of two shells which can be assembled together:

1. Create the housing (1) as a single body, without worrying about the separation.
2. Create a second body (2) separation top, which covers which part of the housing will become the top half of the housing
3. Create a bottom separation (3) as a third body in the same way
4. Create the top housing as a boolean operation between the housing and the top separation
5. Create the bottom housing as a boolean operation between the housing and the bottom separation

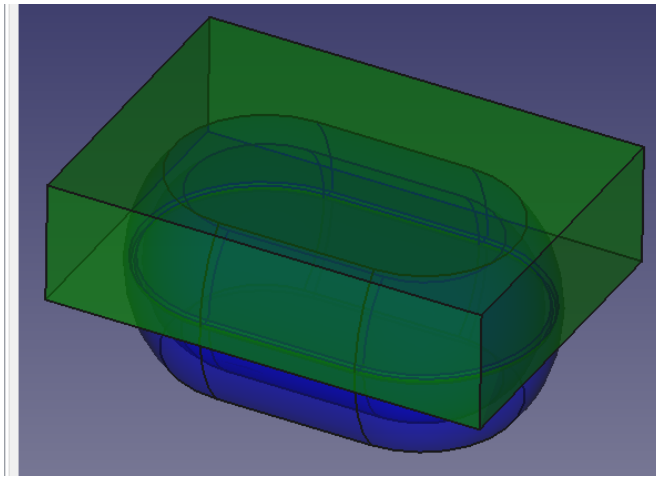


Create the first body named **Housing** (1). In this example it consists of an additive loft for the outside and a subtractive loft for the internal cavity.



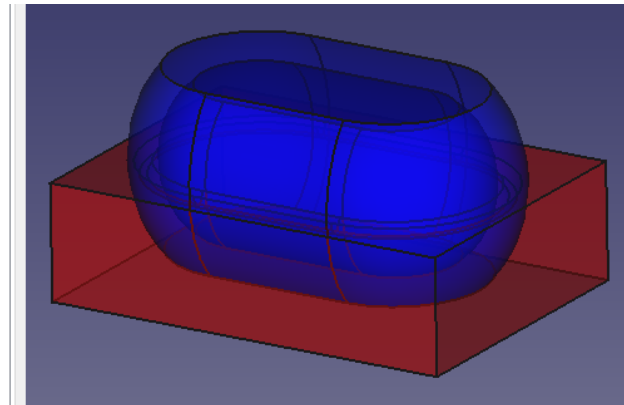
Then create another body in the same part, and call it **Separation top** (2). In this example, it consists of a pad and an additive pipe to create the rim. Note that the pad is deliberately larger than the housing: since we will create a boolean operation later on, the exact size does not matter.

- ▼ Housing
 - > Origin
 - > HS Base
 - > HS Cavity
- ▼ **Separation top**
 - > Origin001
 - > ST Base
 - > ST Rim

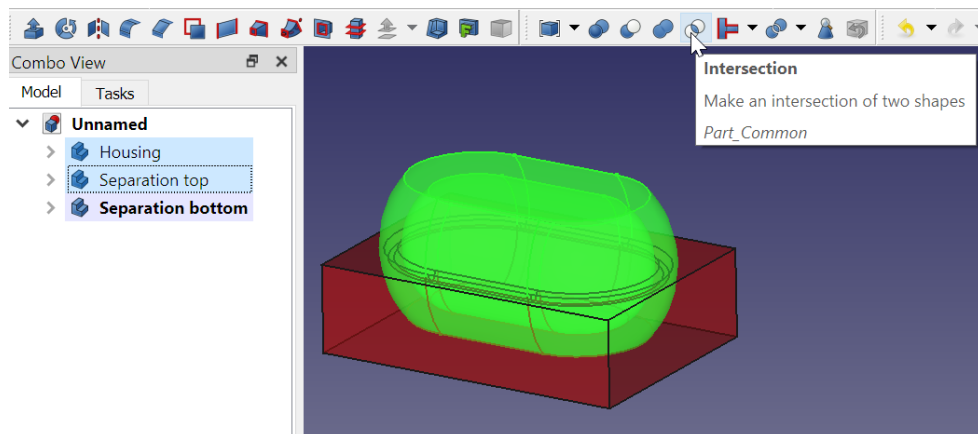


Also create **Separation bottom** (3) as a separate body.

- > Housing
- > Separation top
- ▼ **Separation bottom**
 - > Origin002
 - ▼ SB Base
 - sb base
 - > SB Groove

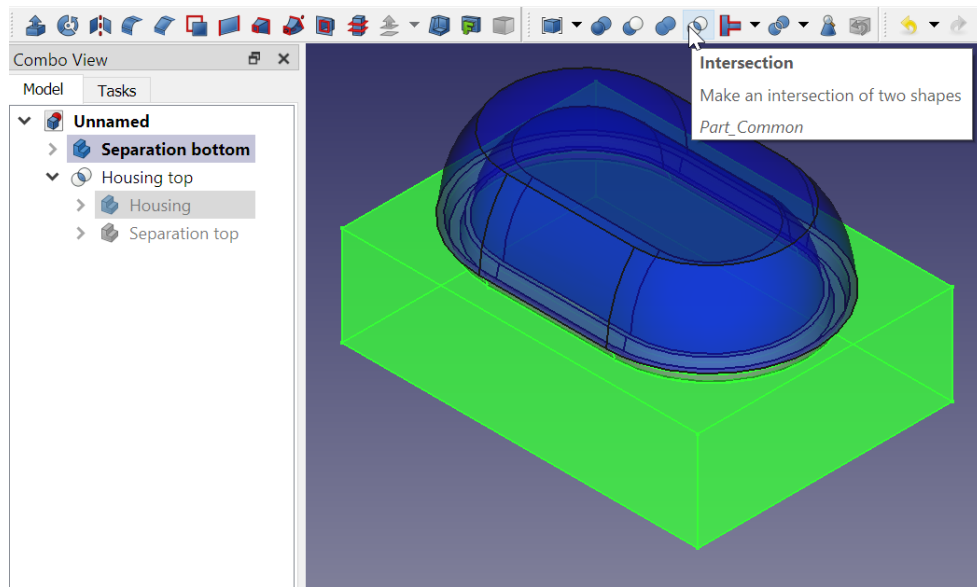


Next move to the part workbench. Select **Housing** and **Separation top**, and choose the **Intersection** command from the toolbar.



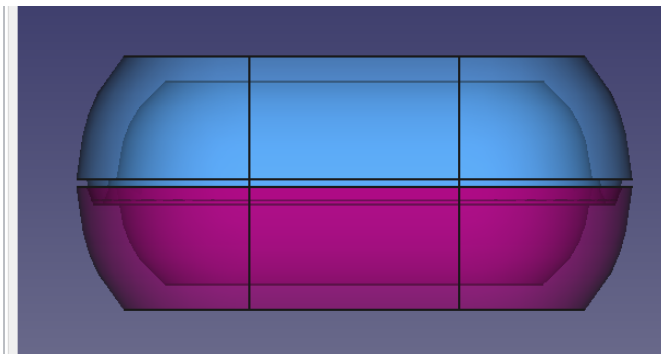
Thus a new body named **Common** is created, consisting of the boolean intersection of both selected bodies. Rename that body **Housing top** (1 & 2).

It may seem as if the **Housing** body has disappeared from the model tree, so we can no longer select it to create the bottom housing. However, if we expand the **Housing top** body, we can see that **Housing** is still there since it was used to make up the **Housing top**. Select the **Housing** body and the **Separation bottom** body and apply the **Intersection** command again on those two bodies. The resulting body is named **Common 001**. Rename it to **Housing bottom** (1 & 3).



After modifying the color and transparency of both housing parts, the result looks like this:

- ▼ Housing top
 - > Housing
 - > Separation top
- ▼ Housing bottom
 - > Housing
 - > Separation bottom



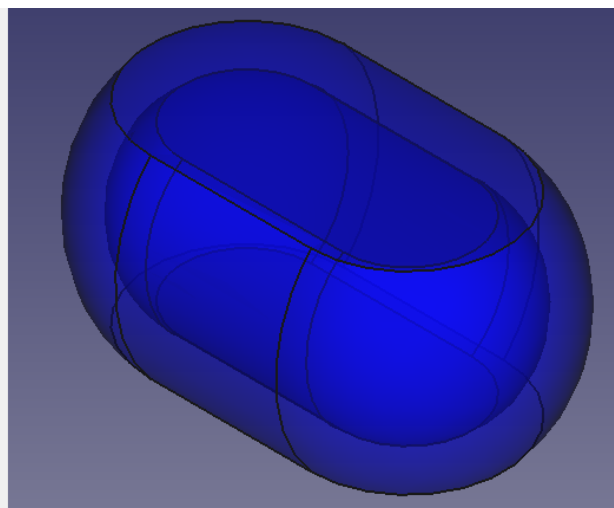
Making changes to the housing

One could argue that the drawback of this method is that we can no longer modify both **Housing top** and **Housing bottom** using the part design workbench. In practice this is not a problem, since we can make those modifications to the three bodies we started with. However, it is important to make a considerate decision about on which body to make the modification. For instance, if the housing is used to support some electronics, and we need a power cable to connect to the electronics inside, we can simply add those features to the original **Housing** body.

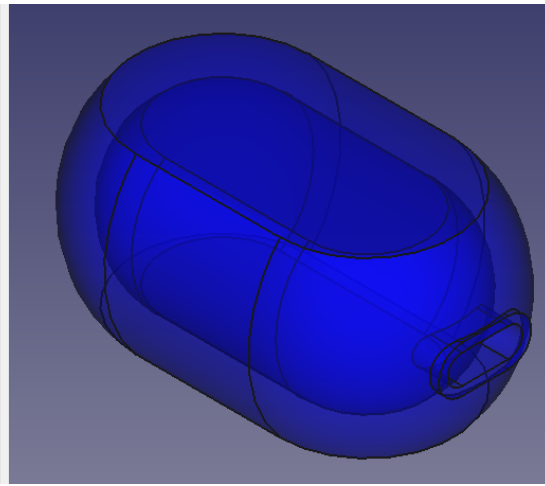
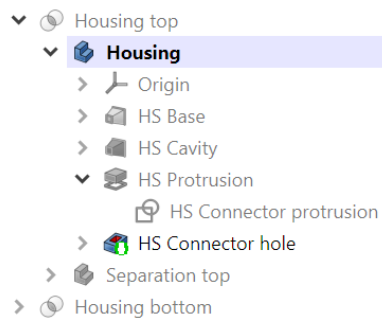
Ensure that **Housing** is the only visible body, and that it is the active part by doubleclicking it.

- ▼ Housing top
- ▼ **Housing**
 - > Origin
 - > HS Base
 - > HS Cavity
 - > Separation top
- ▼ Housing bottom

Property	Value



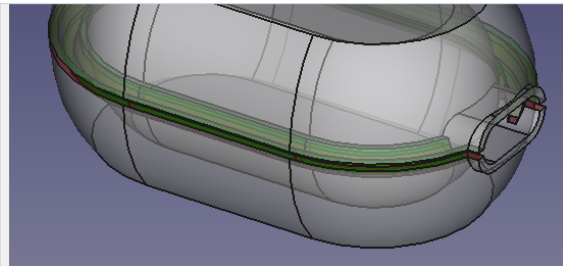
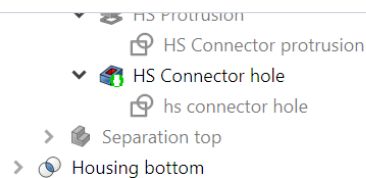
Next, add a protrusion and a hole to the housing for the power connector.



After making the **Housing** body invisible and the **Housing bottom** and **Housing top** bodies visible, we can see that both bodies were modified:



README.md

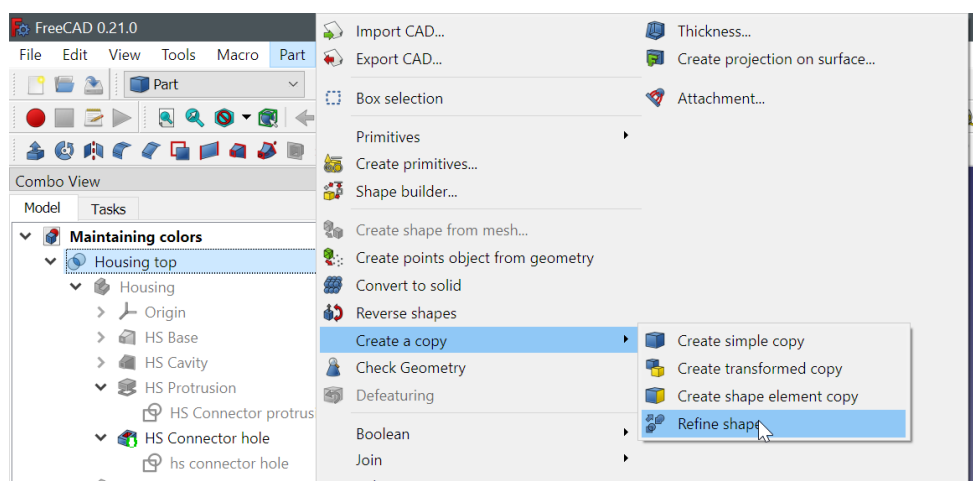


Please note that this operation has overwritten the colors of the **Housing bottom** and **Housing top** bodies. We can again correct these colors, but this soon becomes annoying and so we apply a simple workaround to avoid that.





Maintaining the colors of both housing bodies

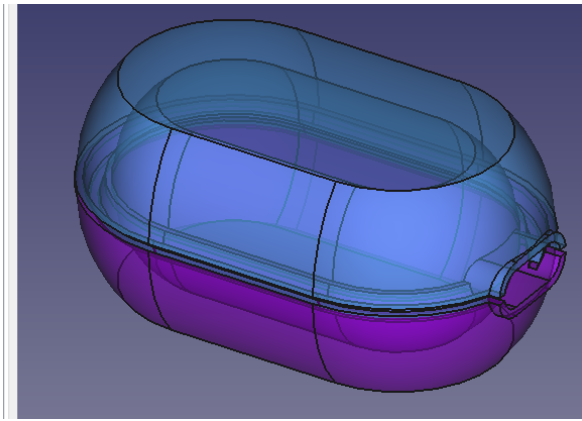
The simplest workaround is to create a copy of the bodies and apply the desired color to this copy.

Switch to the **Part** workbench and select the **Housing top** body.








Rename the copied body **Housing top refined** and modify color and transparency. Repeat this for **Housing bottom**.

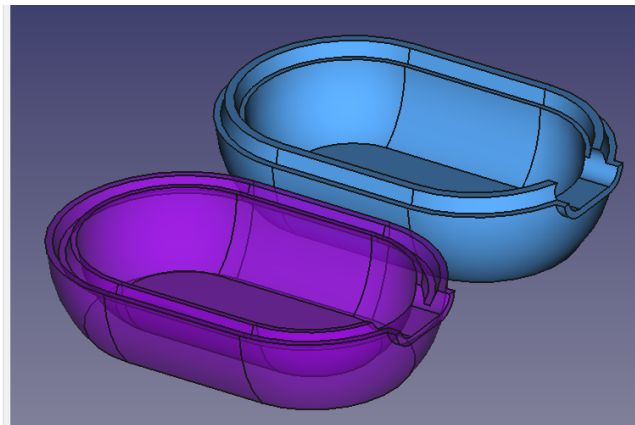
- >  Housing top
- >  Housing bottom
-  Housing top refined
-  Housing bottom refined



The colors of the refined shapes will remain unchanged if the original bodies are modified.

Another advantage of this workaround is that we can create multiple copies in various positions. This way we can easily inspect the parts in different orientations by making the right combination visible.

- >  Housing top
- >  Housing bottom
-  Housing top closed
-  Housing bottom refined
-  Housing top opened



Applying a naming convention for bodies and features

If the number of bodies and features in the model tree grows, it becomes increasingly difficult to identify the right feature if you want to make changes. It becomes helpful to choose unique and meaningful names for features. I have developed my own system which is as follows:

Type of feature	Case	Code
Volumetric features	Sentence case	Bd Name
Non-volumetric features	lowercase	bd typ name ext

Non-volumetric features such as sketches and planes use lower case names. Volumes use Sentence case names (e.g., a sketch may be named **hs base**, and the pad that is created using that sketch is named **HS Base**)

Code	Meaning
Bd	a loose abbreviation of the body (HS for housing, ST for separation top, SB for separation bottom, etc.)
Typ	a 3 letter code for the type of feature after that abbreviation: pln for a plane axs for an axis ref for a shape binder
Name	The name of the feature
ext	for additive and subtractive pipes: the trajectory of the pipe is followed by trj the cross section is followed by crs

Examples:

Name of the feature	Purpose
sk top	Top view of the part in the Skeleton body
hs ref top	Shape binder in the housing body, referencing the top view in the Skeleton body
hs pln bottom	Datum plane in the housing body, representing the bottom of the Housing body
sb groove crs	Cross section of the groove in the Separation bottom body
sb groove trj	Trajectory of the groove in the Separation bottom body
SB Groove	The 3D groove in the Separation bottom body, made up of sb groove crs and sb groove trj

It is helpful to choose a pragmatic approach: for simple projects, the overhead of renaming every feature may not be worth the effort.

Using a skeleton to drive dimensions of the bodies

As can be seen in this example, it would be helpful to create links between the different bodies to make the design truly parametric. For instance, the rim is defined in the **Separation top** and **Separation bottom** bodies, but they need to follow the contour that is defined in the **Housing** body. One body can reference another body, but the reference can only be in one direction: once features of a body B are referencing body A, there can no longer be a reference from body A to body B.

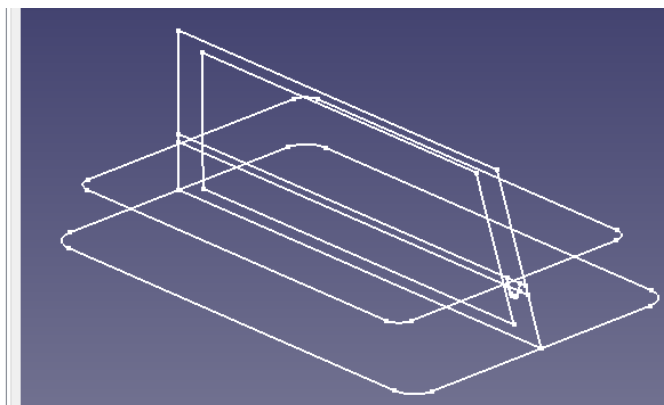
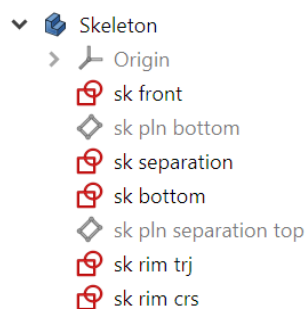
One way of keeping this structured is to start with a **Skeleton** body which only holds some basic shapes and dimensions, but that does not represent any volumes. This **Skeleton** body is then referenced by the other bodies.

Another advantage of a skeleton body is that it makes the model more robust. If sketches refer to 3D geometry, such as edges of the body, the model quickly becomes unstable since names of those edges are changed when making small changes (the notorious [Topological Naming Problem](#)). When referring to edges in sketches instead, it is less likely that names of those edges are changed. This is especially true if we keep these sketches small and simple. It is therefore better to create a large number of simple sketches instead of a few complex ones.

In this example, the **Skeleton** body contains a number of sketches:

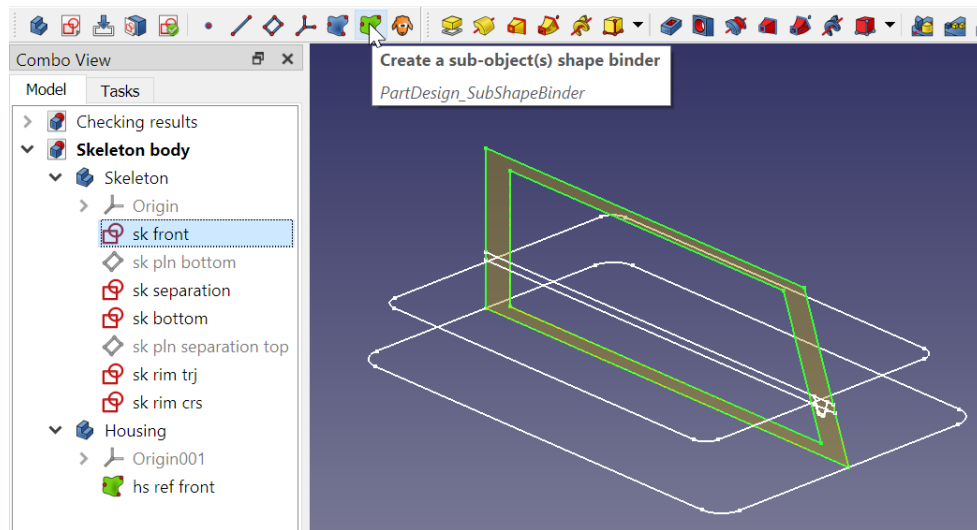
- **sk front**: the front view of the housing
- **sk separation**: the separation lines for both the top and the bottom separations
- **sk bottom**: the bottom view of the housing
- **sk rim trj**: the trajectory that the rim and the groove must follow
- **sk rim crs**: the cross sections of both the rim and the groove

Besides, it also contains some helper planes that were used to create these sketches. The sketches also refer to each other: for instance, the length of the housing is both defined in **sk front** and in **sk bottom**. Therefore, **sk bottom** refers to **sk front** to obtain the length so the length is defined only once.



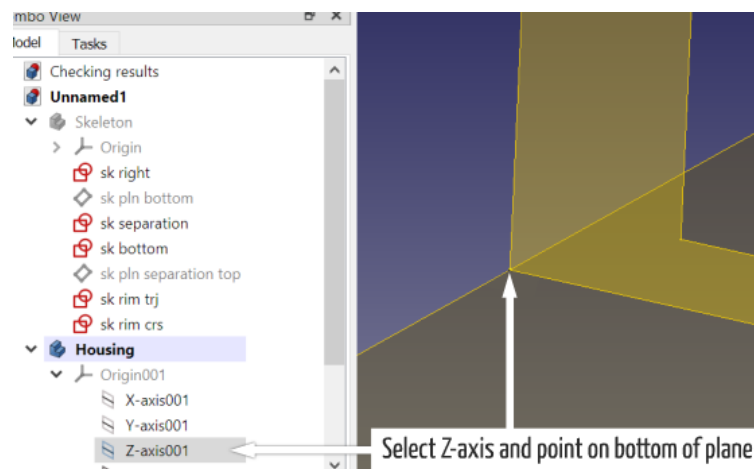
Now create a second body with the name **Housing**. It is not possible to create links to sketches in other bodies like we are used to. We first need to create a *Shape binder*:

- Ensure that **Housing** is the active body
- Select the **sk front** sketch in the **Skeleton** body
- Use the Create a sub-object(s) shape binder -button on the toolbar to create a shape binder
- Rename the shape binder **hs ref front** (I will explain the name convention later)

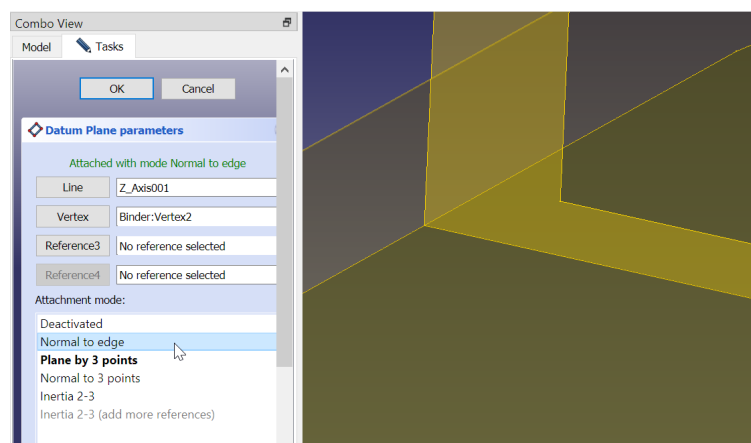


Repeat the procedure for **sk bottom** and rename it **hs ref bottom**. Now make the **Skeleton** body invisible using the spacebar. Then create a plane where the bottom sketch will be drawn:

1. Select the Z-axis of the **Housing** body in the model tree,
2. Also select a point at the bottom of the housing in the model.

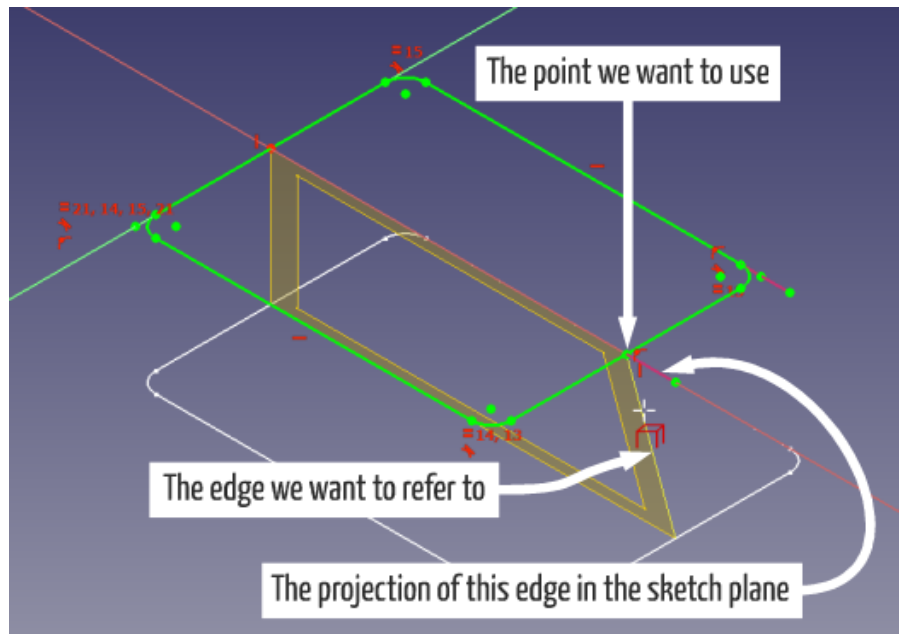


3. Now create a plane using the **Datum Plane** button on the toolbar
4. Select 'normal to edge' in the Datum plane parameter window and click OK

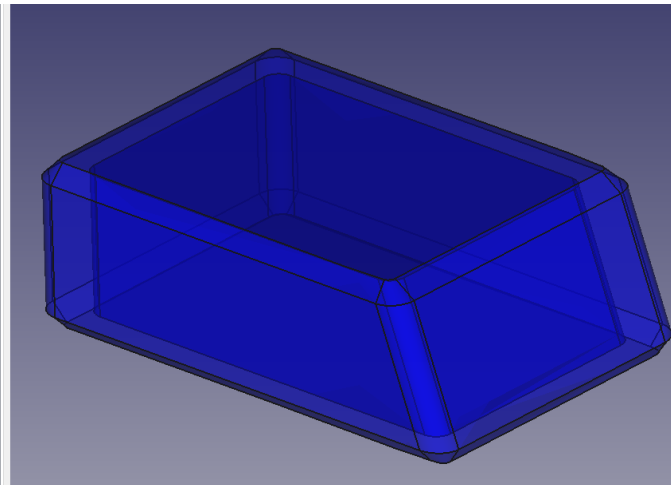
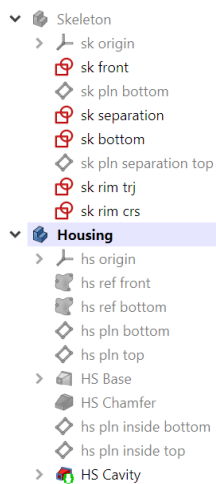


We now have a plane that we can use for the bottom sketch. Now create the bottom sketch of the housing by simply tracing the **hs ref bottom** shape binder.

Repeat the same procedure with the top plane. It is possible to reference the oblique edge of the front view by changing the sketch view to isometric and selecting the oblique edge



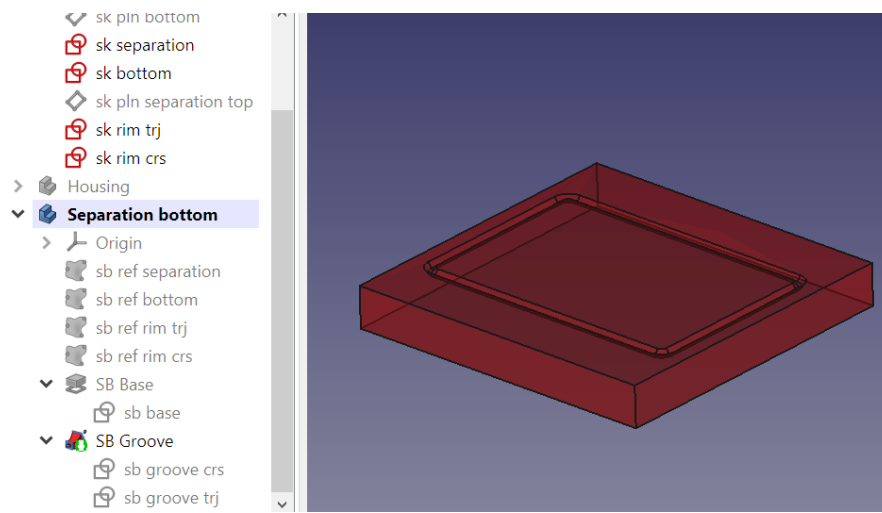
A chamfer is added to the top and bottom edges of the housing. The cavity inside the housing is created in a similar way as the outer shape.



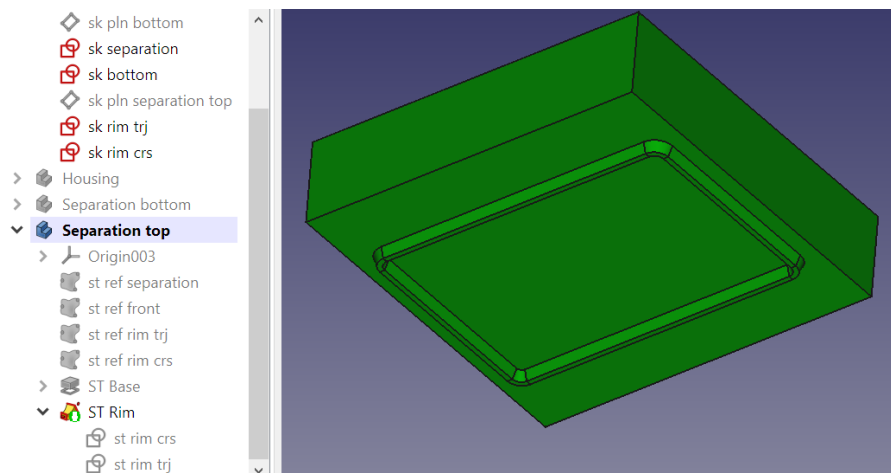
Note that:

- the main outer dimensions of the housing can be changed by only changing dimensions in the **Skeleton** body
- not all sketches from the **Skeleton** body have been imported, e.g. the rim is not needed in the **Housing** body
- details which are independent from other bodies (such as the chamfer), were only defined in the **Housing** body

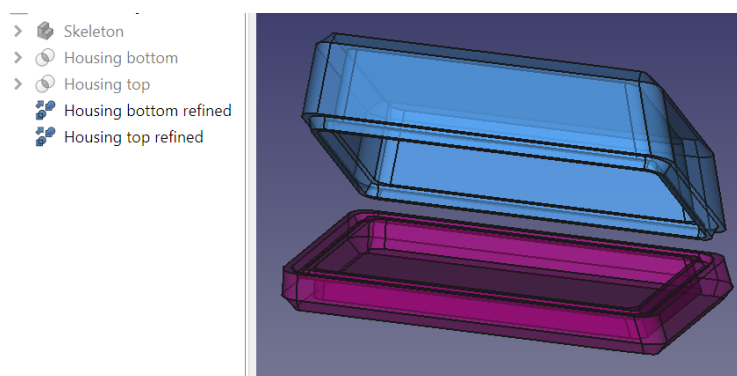
The **Separation bottom** body is created in a similar way:



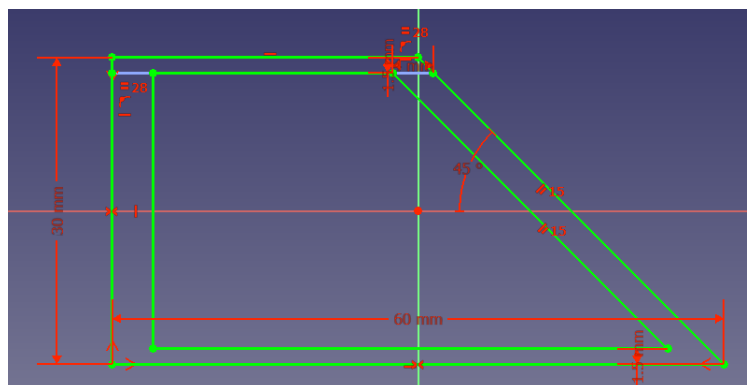
As goes for Separation top:



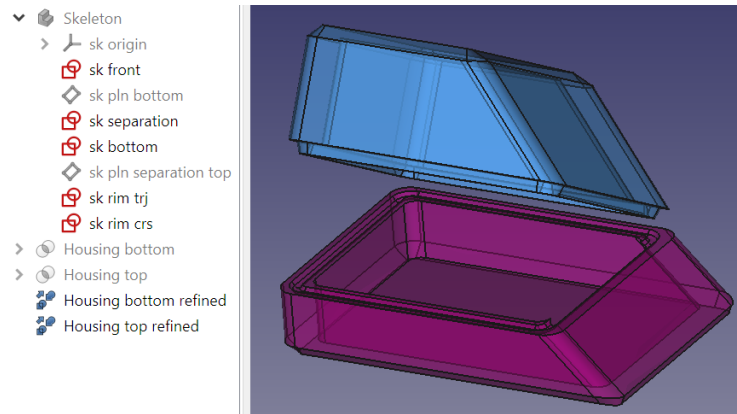
With boolean operations and refined shapes, both halves look like:



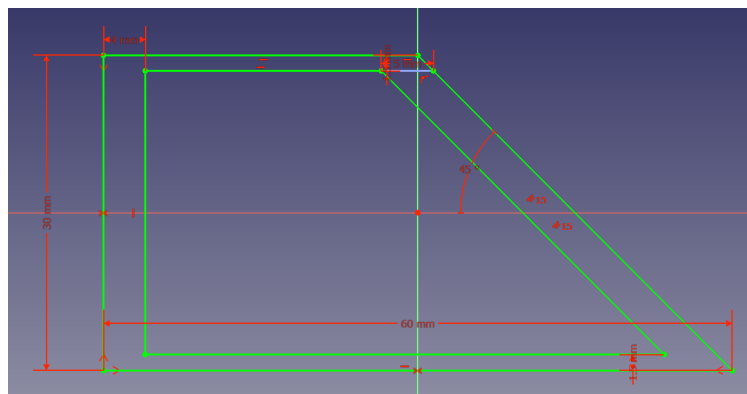
The proof of the pudding is in the eating. We change a few dimensions in the **sk front** sketch in the **Skeleton** body to see if the model is indeed parametric:



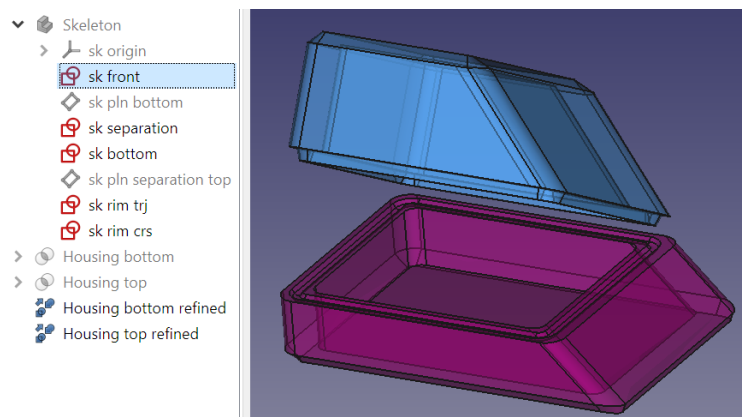
Indeed, the result is as expected:



It appears that the wall of the bottom part is too thin to support the groove, so the inner wall of the groove is no longer there. The correction for this can be made in the same sketch:



This effectively fixes the groove:



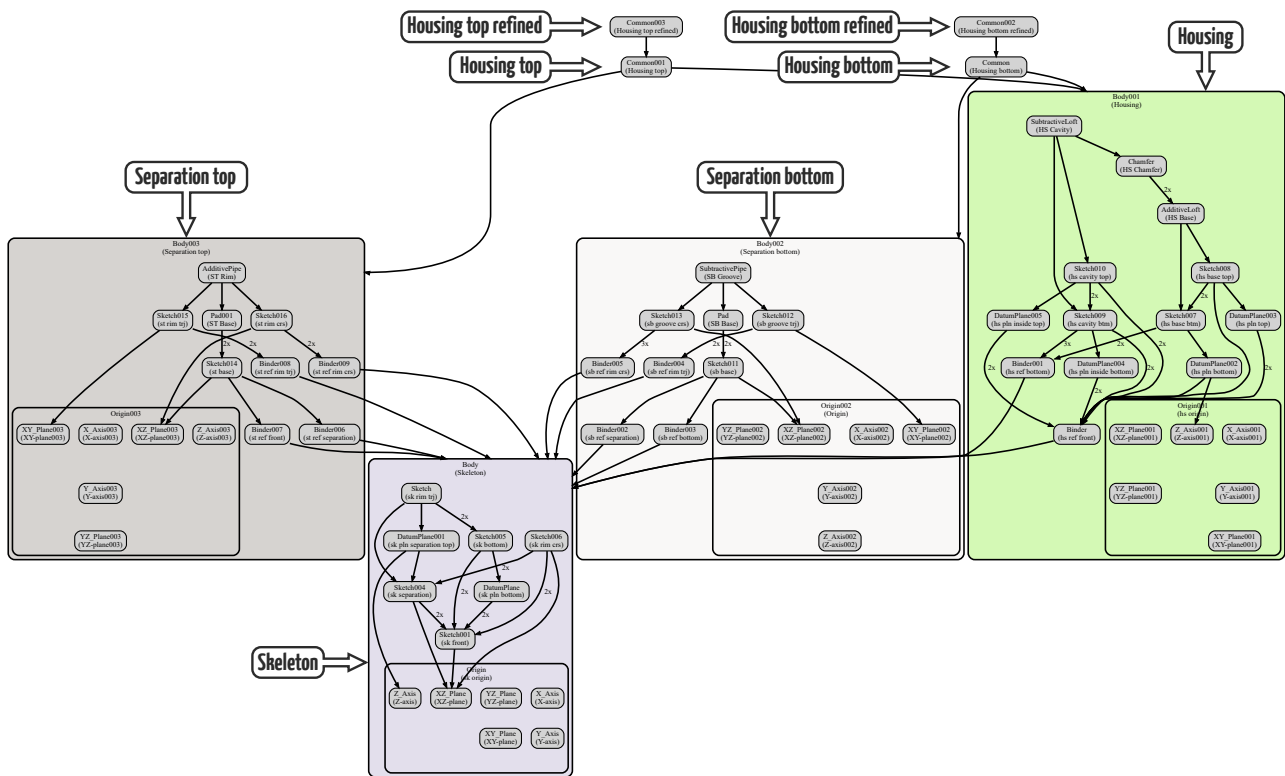
Checking the model

Checking links

It can sometimes (although rarely) occur that links between bodies cause errors that are very hard to find. Sometimes the problem is that there are crosslinks between bodies, i.e. body A refers to body B and body B refers back to body A. This circular reference causes FreeCAD to stop automatic recalculation of the part.

The dependency graph (menu Tools > Dependency Graph) can be very helpful to spot those errors. To use this tool, the third party software [Graphviz](https://wiki.freecad.org/Std_DependencyGraph) needs to be installed (see https://wiki.freecad.org/Std_DependencyGraph).


The dependency graph of the housing looks like this (text balloons were added manually):



The graph shows that:

- Bodies **Separation Top**, **Separation Bottom** and **Housing** all refer to the **Skeleton** body
- Body **Housing top** refers to **Separation top** and **Housing**
- Body **Housing bottom** refers to **Separation bottom** and **Housing**
- Bodies **Housing bottom refined** and **Housing top refined** refer to **Housing bottom** and **Housing top** respectively
- References made by the **Part** workbench act on bodies, while references made by the **Part design** workbench act on features
- All arrows between the parts are black, indicating there are no errors in this graph

Using the Check geometry tool

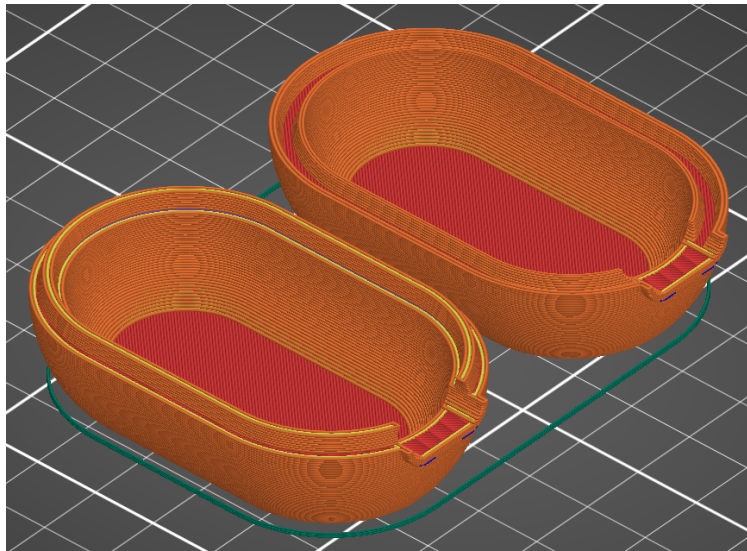
The Check geometry tool from the part workbench can be used to check if the 3D model is valid (Part workbench > Part > Check geometry ). It is beyond the scope of this tutorial to explain how to solve common problems. MangoJelly has an [excellent video](#) on this tool. If causes are hard to find, the FreeCAD community is also willing to help out.

Checking the result in the slicer

I'm using this technique often for 3D printing projects. One of the lessons I learned the hard way is that it is important to regularly check if the parts are printable.

Things to specifically pay attention to:

- are all details still large enough to print?
- would a different orientation of the separation plane make printing easier?
- is it possible to avoid support structures easily?
- is it possible to reduce print time by making other design choices?



As can be seen in this screenshot, both the top of the rim and the sides of the the groove are printale with multiple adjacent tracks. The dark blue lines indicate that the protrusion around the power connector is partially unsupported, but since these areas are very small, we will probably be fine.

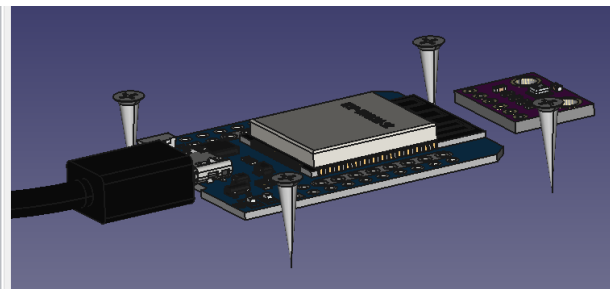
Creating references to the internal components of the housing

In the next example, we will build a housing for an internet of things application. The device will contain a thermometer/barometer/hygrometer connected to a microcontroller. The microcontroller can record the environmental conditions and report logging information over a wireless link.

For projects like this, it is important to obtain accurate 3D models. Usually they are available as STEP file or in another format which can be imported in FreeCAD.

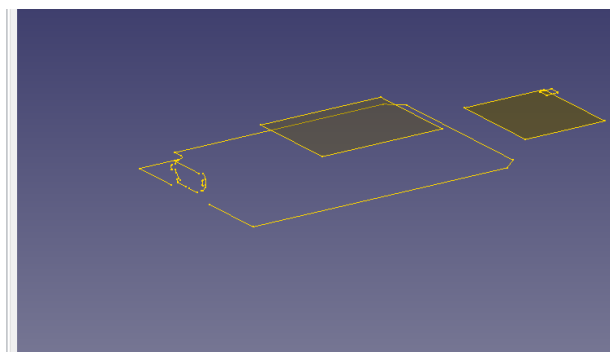
Import the electronic components in the FreeCAD file and orient them well. In this project there is a risk that the heat of the wifi module of the microcontroller affects the temperature measurement of the sensor, so it is important to minimize thermal crosstalk when designing the housing.

- > GY-BME/P280, 6-PIN-Version v4
- > MH-ET_LIVE_MiniKit
- > Micro USB connector
- > SelftappingScrew2.6x12_01
- > SelftappingScrew2.6x12_002
- > SelftappingScrew2.6x12_003
- > SelftappingScrew2.6x12_004

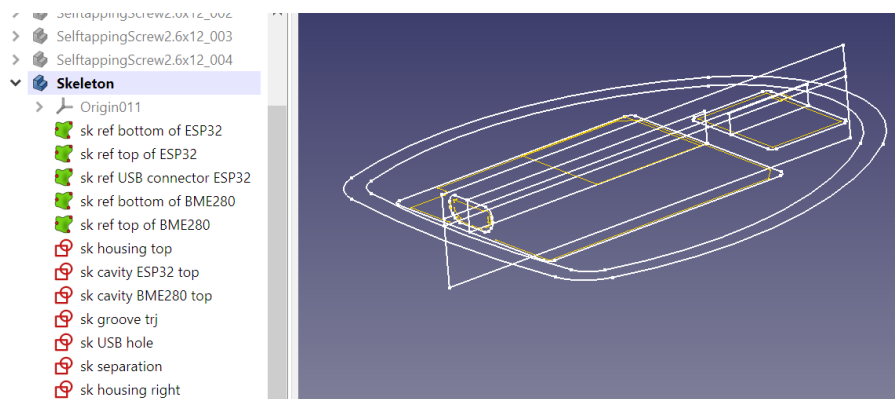


Create the **Skeleton** body. In the skeleton, import important geometry of the components using shape binders. This way, the sketches in the skeleton will dynamically follow the components when the components are moved.

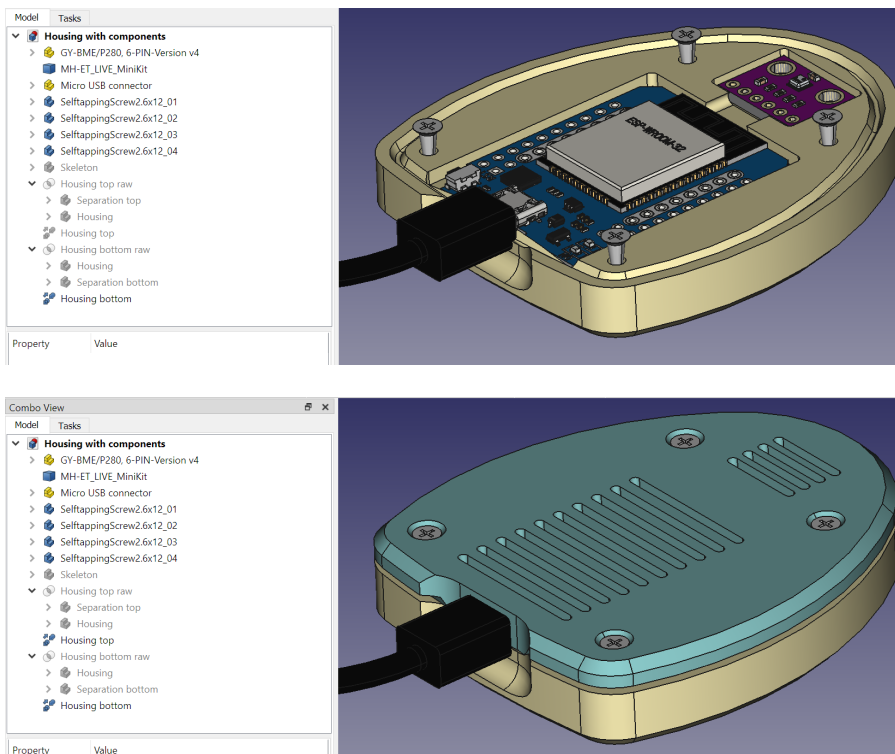
- > GY-BME/P280, 6-PIN-Version v4
- > MH-ET_LIVE_MiniKit
- > Micro USB connector
- > SelftappingScrew2.6x12_01
- > SelftappingScrew2.6x12_002
- > SelftappingScrew2.6x12_003
- > SelftappingScrew2.6x12_004
- ▼ **Skeleton**
 - > Origin011
 - > sk ref bottom of ESP32
 - > sk ref top of ESP32
 - > sk ref USB connector ESP
 - > sk ref bottom of BME280
 - > sk ref top of BME280



Create the sketches in the **Skeleton** body.



Create the Housing, Separation top, Separation bottom, Housing top and Housing bottom like in the previous example.



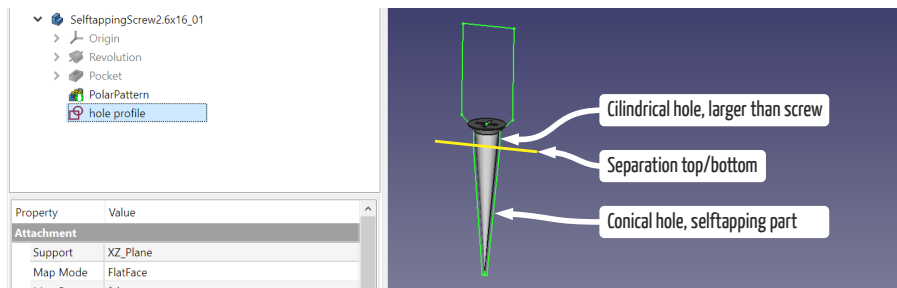
We can now move the boards around, and (within certain limits), the cavities in the housing will follow the components.

Using self tapping screws to close the housing

I often use self tapping screws for these housings. With the right tolerances, these screws work really well and require no post processing (tapping, inserts) in the parts, which makes it quite fast. These screws are available from many different suppliers at AliExpress.



In order to make the screw holes parametric, I created a model of the screw which contains an additional sketch representing the hole in the housing.



In reality, these screws are not conical. Some day I will make a model that more closely resembles the shape of these screws. Nonetheless, this shape works in my printed models.



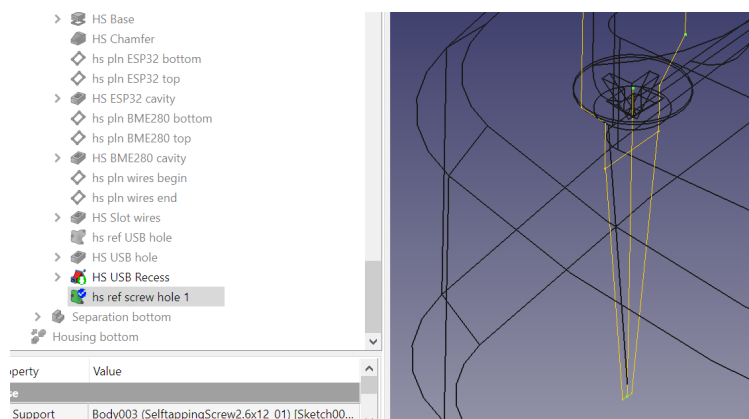
By tweaking the shape of the hole, I achieved a good fit for these screws for PET printed on my Prusa Mk3s. With a different printer or material the tweaking may have a different outcome.

Creating a screw hole

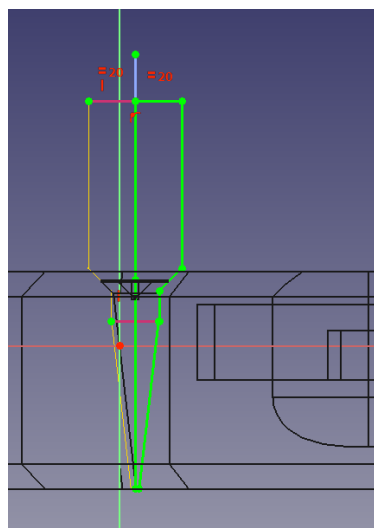
This is how it works:

Insert screw in the model using File > Merge project

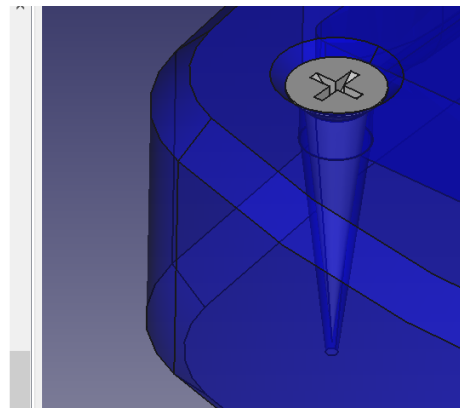
1. Create a shape binder **hs ref screw hole 1** in the **Housing** body, referencing the screw hole sketch from the model of the screw (no need to make an intermediate reference in the **Skeleton** body)
2. Make the model of the original screw invisible (so we can only select elements from the shape binder)
3. Select three points on the shape binder of the hole, and create a datum plane **hs pln screw hole 1** through these points



4. Create a sketch **hs screw hole 1** on this datum plane, tracing one half of the screw hole
5. Add a construction geometry line to this sketch, representing the centerline of the screw. I usually make the length equal to an arbitrary other line of the sketch to make the sketch fully defined.



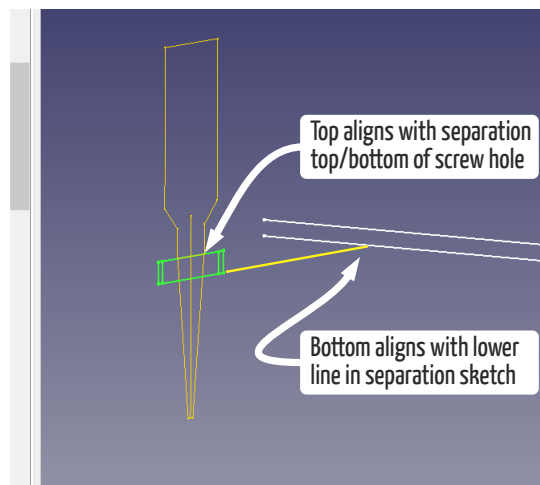
6. Create a Groove **HS Screw hole 1** based on this sketch, choosing the construction line as a centerline.



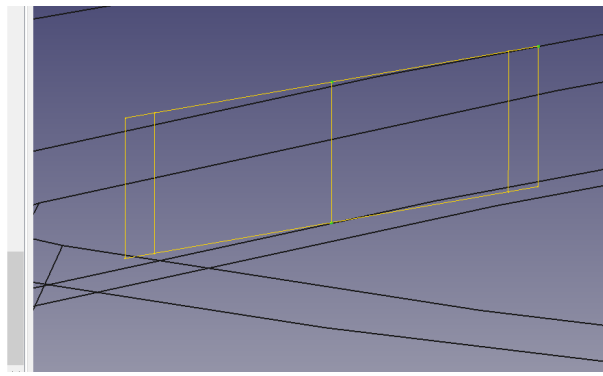
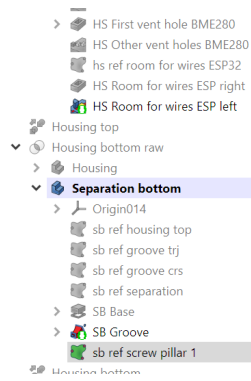
Creating a pillar for the screw

To solve this, we can make a local pillar in the bottom housing and a hole in the top housing.

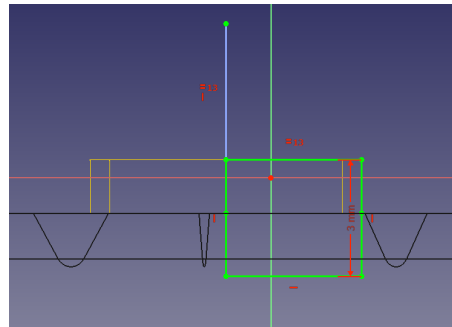
- ↳ **Skeleton**
 - ↳ Origin011
 - sk ref bottom of ESP32
 - sk ref top of ESP32
 - sk ref USB connector ESP32
 - sk ref bottom of BME280
 - sk ref top of BME280
 - sk ref wires ESP32
 - sk housing top
 - sk cavity ESP32 top
 - sk cavity BME280 top
 - sk groove trj
 - sk groove crs
 - sk USB hole
 - sk separation
 - sk housing right
 - sk room for wires ESP32
 - sk ref screw hole 1
 - sk pln screw hole 1
 - sk screw pillar 1



5. Make **Separation bottom** the active body
6. Create a shape binder of **sk screw pillar 1** from the **Skeleton** body and rename it **sb ref screw pillar 1**
7. Create a datum plane **sb pln screw pillar 1** like we did in the housing



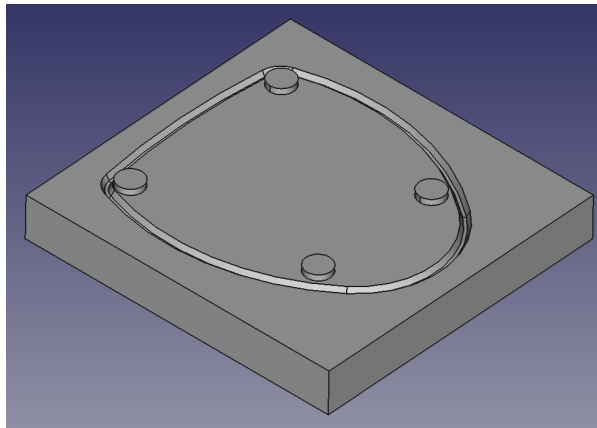
8. Create a sketch **sb screw pillar 1** to create the pillar, and add a geometry line that will be the center line of the pillar



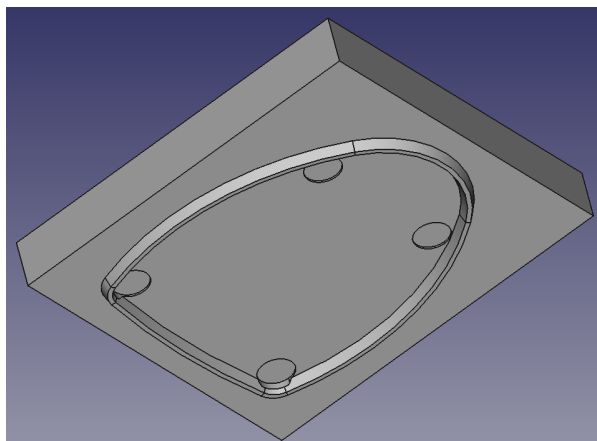
In this case, I extended the bottom of the pillar so it would also fit the V-groove. Also, do not forget the centerline.

9. Create a revolution and name it **SB Screw pillar 1**

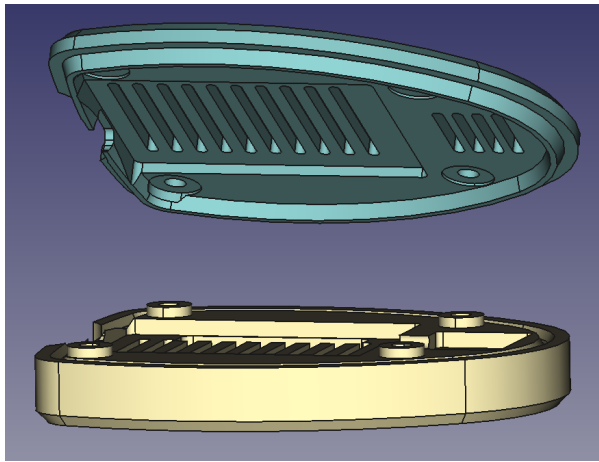
10. Repeat the same procedure for the other pillars. In this case, I did pillar 3 in the same way and created pillars 2 and 4 by mirroring.



Also repeat the procedure to create the holes in **Separation top**.



The changes will now automatically come through in both housing parts:



This is a good example to demonstrate that some changes need modifications in the housing part, while others require changes in the separation parts.

Creating a complex hinge

Creating references to external parts

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages

● Python 100.0%

Suggested Workflows

Based on your tech stack



Actions Importer

Automatically convert CI/CD files to YAML for GitHub Actions.

Set up



Pylint

Lint a Python application with pylint.

Configure



Publish Python Package

Publish a Python Package to PyPI on release.

Configure

[More workflows](#)

[Dismiss suggestions](#)