3D-Printed Keypad Design

Here is a relatively quick guide into the design and assembly of a 3D printed keyboard. In here, the reader is walked through the design process and the workflow required to build one of these keypads from scratch.

Expected Skills

In this tutorial, you are expected to have these basic skills. However, only the basics are needed

- Programming (C, C++, Java, Arduino)
- Electronics knowledge and soldering (Wiring, soldering, knowledge of Ohm's law, etc.)
- CAD design (TinkerCAD, Fusion 360, etc.)
- Elbow grease

Why design one?

Some reasons you might want to build one for yourself

- Quick access to some keyboard shortcuts like:
  - Ctrl + S (save)
  - Ctrl + Alt + Delete (Task Manager)
  - Win + D (Show desktop/ Panic button)
  - And many more...
- Extra long key sequences for some games, programs
- Open applications with a push of a button.
- Ergonomics (Don't have to stretch your hands across keyboard for shortcuts)

Components/ Bill of Materials

In this tutorial, we consider working with these components:

- Cherry MX switches
- 22 AWG wire
- Sparkfun Pro Micro
- A 3D Printer
- Micro-USB Cable
- Access to the Arduino IDE

Electric Diagram

The keypad boils down to two things: sense whether a button is pressed, and to tell the PC that a character has been entered.

This keypad relies on the concept of pull-up resistors; Sparkfun discusses pull-up resistors in more detail here: https://learn.sparkfun.com/tutorials/pull-up-resistors. We use these pull-up resistors in to allow our controller to determine whether our switch is pressed (pulled low) or released (pulled high). A diagram from Sparkfun quickly illustrates the pull-up circuit:

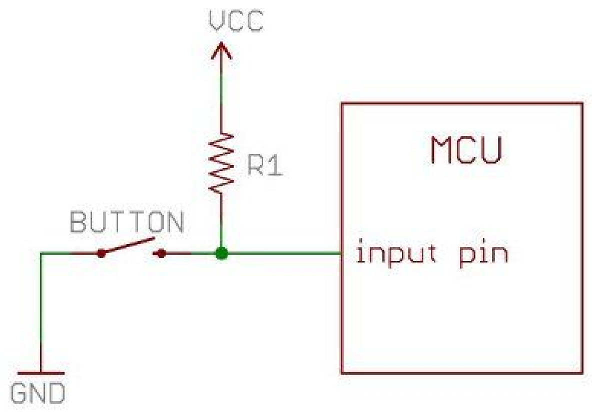

Figure <>.

To quickly explain this circuit, when the BUTTON is pressed, the input pin is connected straight to ground (GND), resulting in the input pin reading 0V. When BUTTON is released, the circuit is disconnected from GND, and by Ohm's Law the input pin sees 5V.

Luckily for us, most Arduino boards have built-in pullup resistors. This means we can wire the buttons straight through to the input pins.

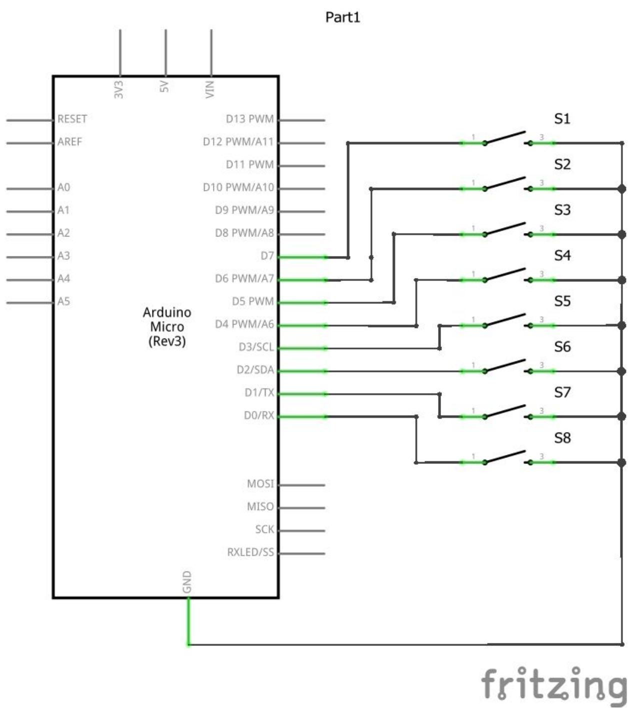

Figure <>.

Physically, this boils down to one red wire per pin, and all black wires connected to one ground pin. On the example below, this is how the buttons should be wired:

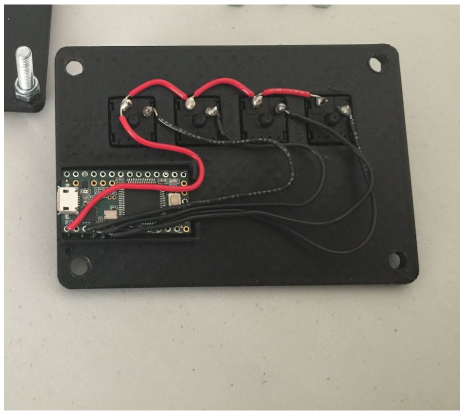

Figure <>.

Using an earlier example, the connections are the same, with the red wire being ground (GND) and the black wires being the inputs to the controller. As long as you keep the colors consistent, you shouldn’t lose track of the GND and input pins

Programming

Here is the code required to program the Pro Micro. We are going to be using the default Arduino keyboard libraries for this. The rest of the instructions are commented in the code. For easier copy/paste, one can find the repository here:
https://github.com/cgarcia2097/DIY-8-Button-Keypad

```
/*
 * Keyboard library usage found here: https://www.arduino.cc/reference/en/language/functions/usb/keyboard/
 * Bounce library used for debouncing found here: https://playground.arduino.cc/Code/Bounce
 * This is done with reference to the Pro Micro and the default Arduino libraries
 * You will need to consult with different libraries if using different boards
 *
 * Thank you Digital Media Experience Labs at Ryerson for helping sponsoring this tutorial,
 * Thank you Thnikk (https://twitter.com/thnikk?lang=en) for helping me create the first prototype
 * Thank you all who have expressed interest in building this keyboard
 *
 */

#include<Keyboard.h>
#include<Bounce2.h>

#define NUM_KEYS 8 // Number of buttons

#define KEY_1 'z'     // Macros go here; characters use quotation marks, modifiers use the ASCII table in
the website below
#define KEY_2 'x'     // Change them here for your convenience
#define KEY_3 'c'     // Modifiers found at: https://www.arduino.cc/en/Reference/KeyboardModifiers
#define KEY_4 'w'     // Pressing multiple buttons at a time requires edits at the loop() sections of the
code
#define KEY_5 'a'
#define KEY_6 's'
#define KEY_7 'd'
#define KEY_8 177

const int key[NUM_KEYS] = {/* Enter the pin numbers of your buttons here, separated by a comma */};
int i = 0;

Bounce button[NUM_KEYS];

void setup() {

  // Start debouncing the switches
  for (i = 0; i < NUM_KEYS; i++) {
    button[i] = Bounce();
    button[i].attach(key[i], INPUT_PULLUP);
    button[i].interval(5);
  }

  // Start your keyboard functionality
  Keyboard.begin();

  // Start serial, to check for your keypresses
  Serial.begin(115200);
  Serial.println("Welcome to Keypad 1.0");
  Serial.println("Scanning buttons now...");
}

void loop() {
  for ( i = 0; i < NUM_KEYS; i++) {
    // Update the button states
    button[i].update();
  }

/*
 * Button #1 - We start checking for keypresses
 * Repeat for the rest of the buttons
 */
  if (button[0].fell()){
    // Press and hold the buttons you want pressed
    Keyboard.press(KEY_1);

    /* If you would like to press more than one keyboard button at a time,
     * repeat Keyboard.press() for the extra set of buttons (up to six due to USB limitations)
     *
     * Like:
     * Keyboard.press(KEY_1);
     * Keyboard.press(KEY_2);
     * Keyboard.press(KEY_3);
     * ...
     * Keyboard.press(KEY_6);
     */

  }
  if (button[0].rose()){
    // Release the same buttons you just pressed
    Keyboard.release(KEY_1);

   /* If you pressed more than one button, you have to release the rest of them
    * Repeat Keyboard.press() for the set of buttons pressed earlier (up to six due to USB limitations)
    *
    * Like:
    *
    * Keyboard.release(KEY_1);
    * Keyboard.release(KEY_2);
    * Keyboard.release(KEY_3);
    * ...
    * Keyboard.release(KEY_6);
    */
  }

/*
 * Button #2 - The second button, do the same as the first
 */
  if (button[1].fell()){
    // Press and hold the buttons you want pressed
    Keyboard.press(KEY_2);
  }
  if (button[1].rose()){
    Keyboard.release(KEY_2);
  }

  // ...
  // ...
  // Keep doing the above until you do all eight

}
```

Cases

What we need to consider?

We need to consider several things when designing a basic keyboard housing. Namely, we worry about the

- **components** (like keyswitches, controller boards, wiring, etc.), **layout** (how our keys and components are laid out onto the case),
- **aesthetic** (shapes, embossed text, etc.),
- **design tools** (hand drawings, CAD modeling, etc) and
- **implementation** (laser cutting, 3D-printing, dremel etc.)

What components are we working with?

In this tutorial, we a working with these components:

- A Sparkfun Pro Micro: https://www.sparkfun.com/products/12640
- Cherry MX switches
- Standard Cherry MX Keycaps

Other components can be used to implement this project, but it is your homework to do the research on those parts. *Not every controller board and switch you see out there can be easily transformed into a keyboard*.

How do we start designing one?

This is where you start drafting the layout of your keyboard.

According to our Cherry MX reference sheethere: https://cdn.sparkfun.com/datasheets/Components/Switches/MX%20Series.pdf we can find methods of mounting our switch. The easiest way to mount our switch of choice is through ***plate mounting***. Plate mounting refers to having our component sit in a hole through some flat object, which is called the *plate*. The diagram on the right is an example of plate mount.

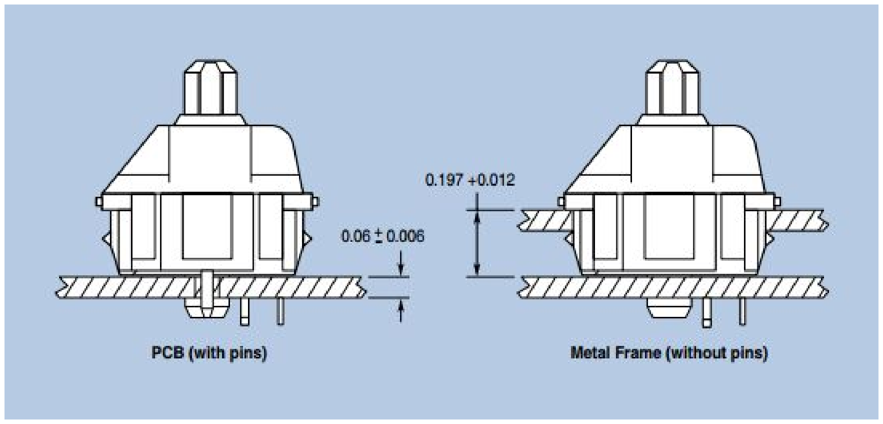

Figure <>. PCB mount on the left, Panel/Plate mount

We shall use plate mount in our design simply because we don't have a printed circuit board (PCB)! A PCB is a board that contains copper tracks that connect electrical components together as if connected together by wires. Since we do not have those at hand, plate mount is essentially our only option if we are going to build this keyboard.

Step 1: Plan and draw the switch holes

According to the Cherry MX switch guide, it says that our switch mounting hole is 14 mm x 14 mm when plate mounted. We must create a 14mm x 14mm square (converted from inches) to accomodate for the switch (MX Series, 2013).

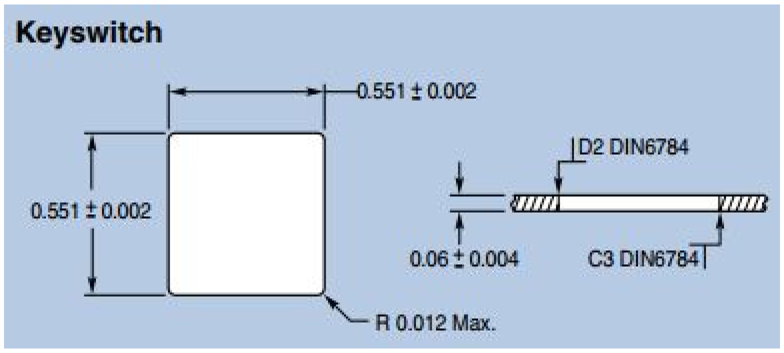

Figure <>. Excerpt from the Cherry MX guide, the mounting hole dimensions

The keycap dimensions however, also must be taken into account. The keycaps are 19mm x 19mm and sit in the center of the 14 mm square.

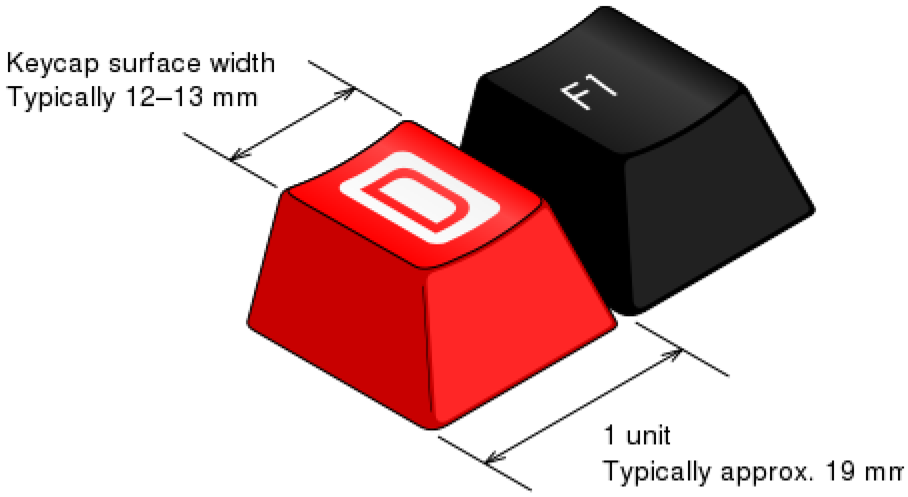

Figure <>.

Therefore, the 14 mm square hole must be positioned inside the 18 mm square, shown below.

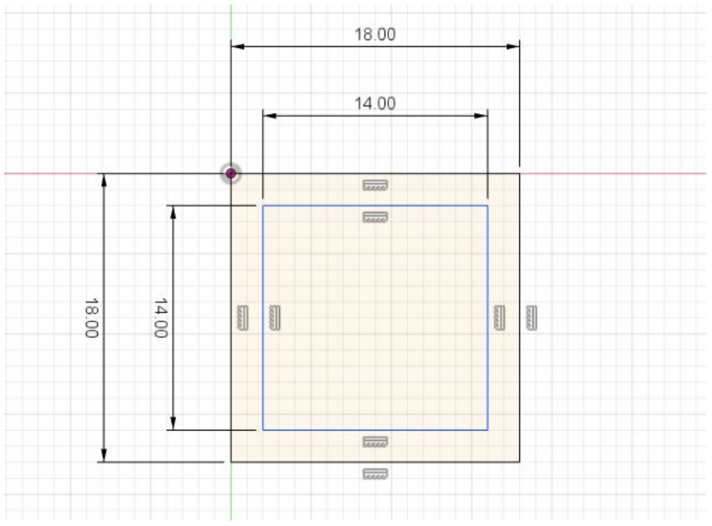

Figure <>. A quick sketch in Fusion 360, small square inside big square.

Rinse and repeat for the number of keys required for your design. In this tutorial, we shall do eight keys total.

Step 2: Creating The Switch/Top Plate

After we have figured out the mounting holes and spacing for the switch, the real creativity begins. However, in this tutorial we shall use a 2x4 grid of keys, giving us 8 keys in total, though your layout may change depending on your taste and number of keys required. This design is done in Autodesk Fusion; as such your methods might differ with different software.

.

1. **We space out the 18 mm squares with 1 mm gaps to give the keycaps some wobble space.** In Fusion, it ends up looking like this:

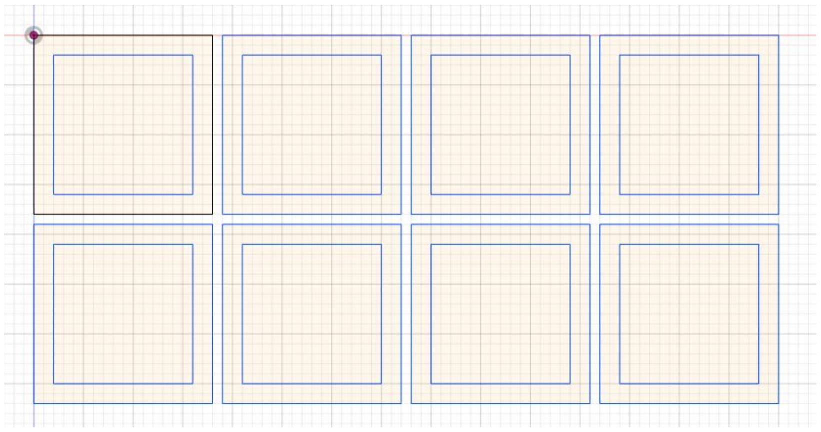

Figure <>.

2. We then pad the edges of these squares with an extra 10 mm of material to cover the entire keypad.

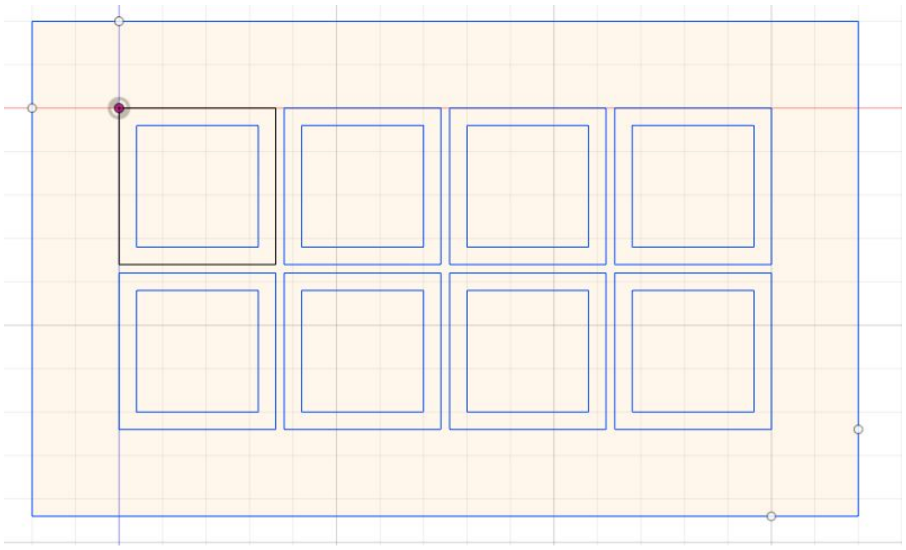

Figure <>.

3. Afterwards, we get rid of the 18 mm squares we used for spacing purposes.

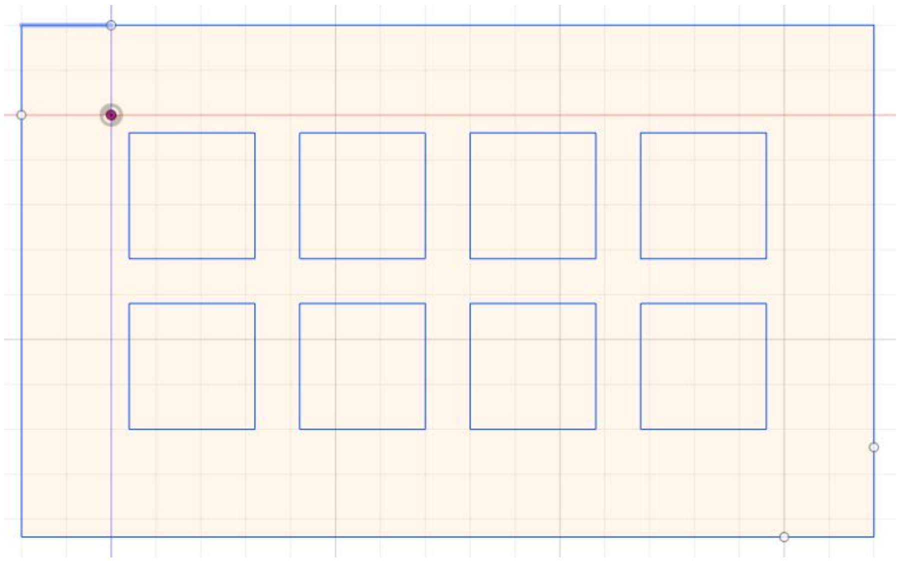

Figure <>.

We have now just finished our reference design. From here, one can get creative with the shapes that enclose the mounting holes for those who are more experienced with CAD design, and can even add embossed logos and text if one desires.

4. We add some rounded corners with radius of 5 mm on each corner:

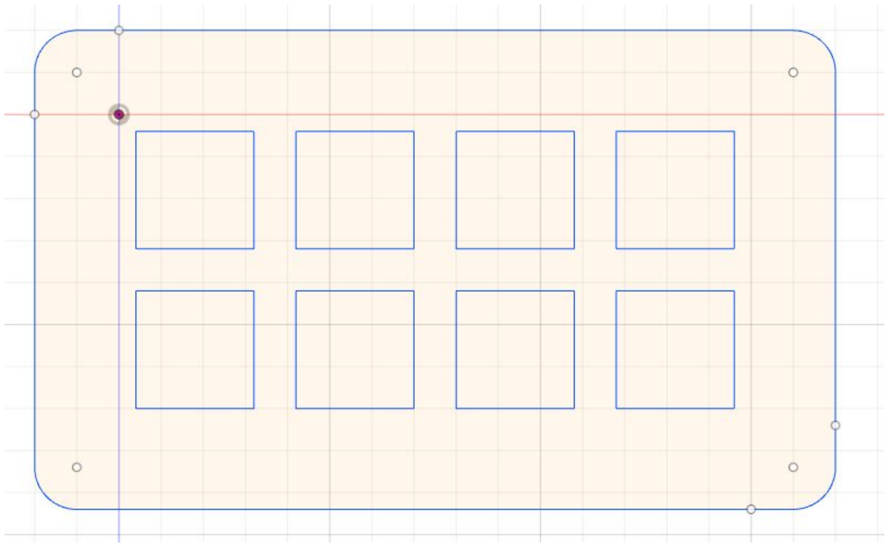

Figure <>.

5. Add some holes of some M3 screws/standoffs on each corner:

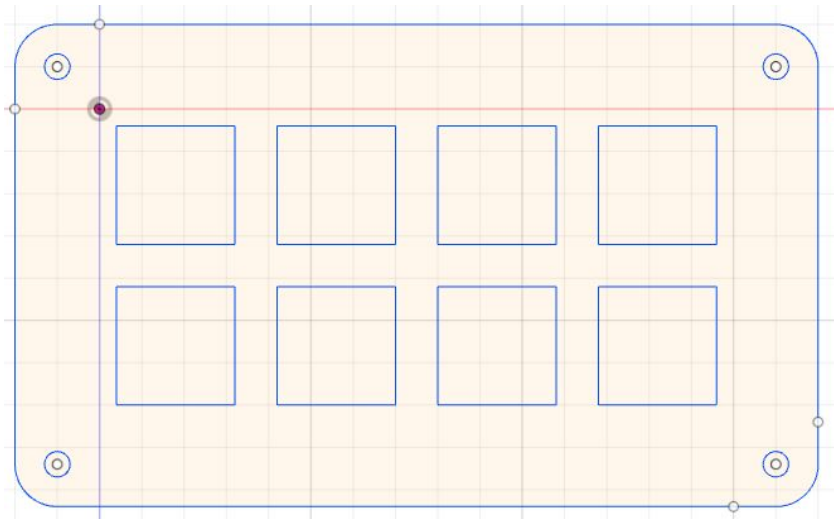

Figure <>.

Voila, your plate is done. Now at this point, one can save this as a DXF, SVG, DWG, etc. and send this to your local laser cutter. However, if you plan to use this as a 3D model, one more step has to be done.

6. **In Fusion, we extrude the resulting face a distance of 3mm**. One can experiment with different thicknesses to one's content if desired.

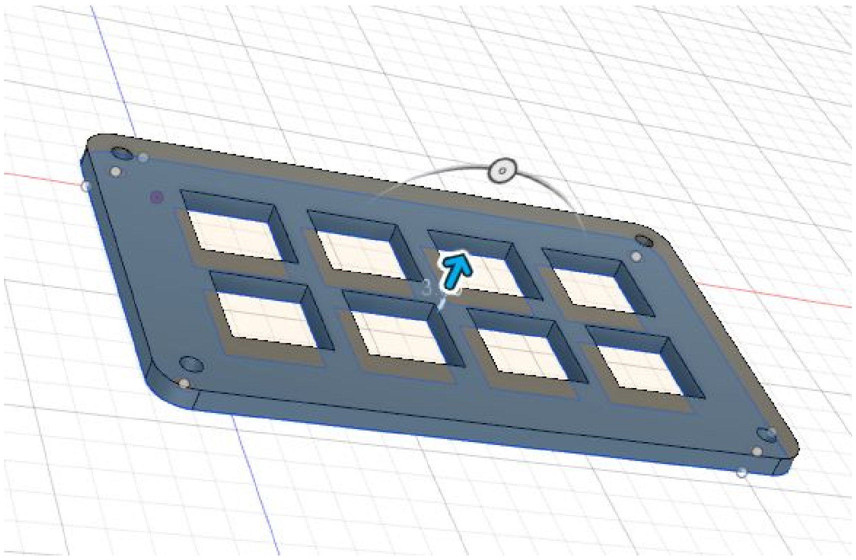

Figure <>.

Voila, your switch plate is finished.

Step 3: The bottom part: Case implementations

Sandwich design

This design method implies the use of 2 flat structures to hold our components together, hence the ‘sandwich’ term used in this design method.

1. **Follow the Plate instructions (up to the 3D modeling part) without creating the holes for our switch, and it should look like this:**

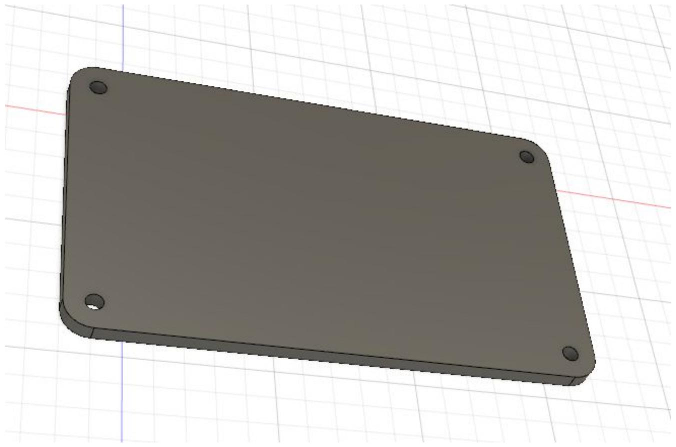

Figure <>.

2. We need to create a 1.3” x 0.7” section to hold our Pro Micro. In this case, it is also preferable if we can get the USB port of the Pro Micro sticking out. We also might need a little bit of wiggle room so make the section a little larger than 1.3” by 0.7”.

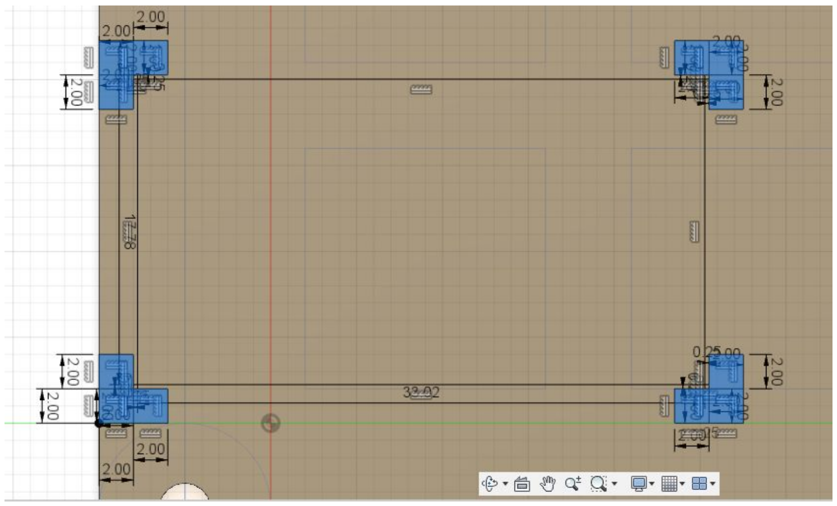

Figure <>.

In the diagram above, the four corners will hold the Pro Micro (the big rectangle in the center) in place (shown by the four L-shaped corners). Note the slight spacing that was added in order to compensate for 3D printing and slightly-off spec dimensions of the PCB (could be slightly larger or smaller).

3. We then extrude the selected corners by 3 mm, and both plates are done:

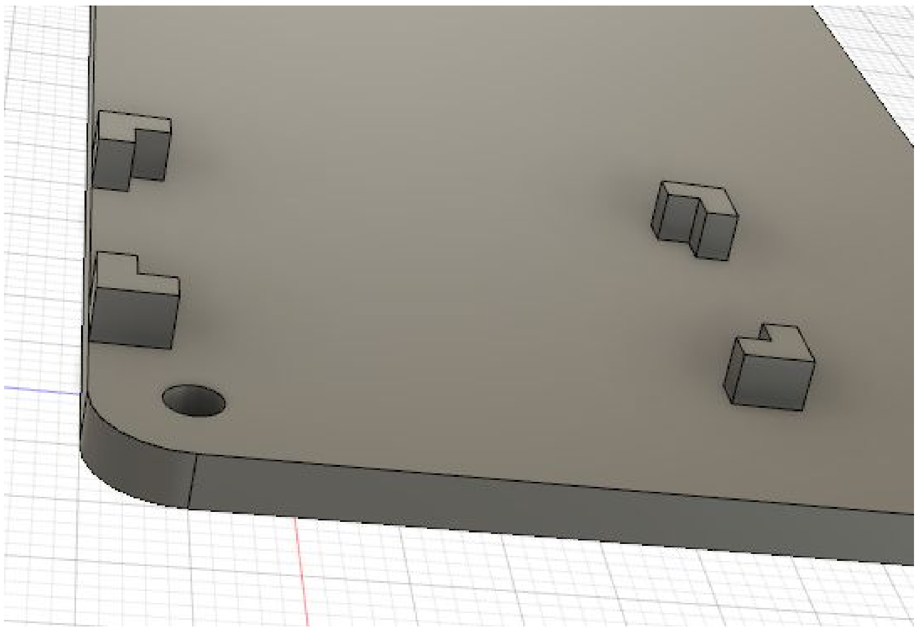

Figure <>.

Our finished sandwich design looks like this. In assembly, you also must purchase M3 standoffs in order to assemble the sandwich case, and optionally, some rubber feet:

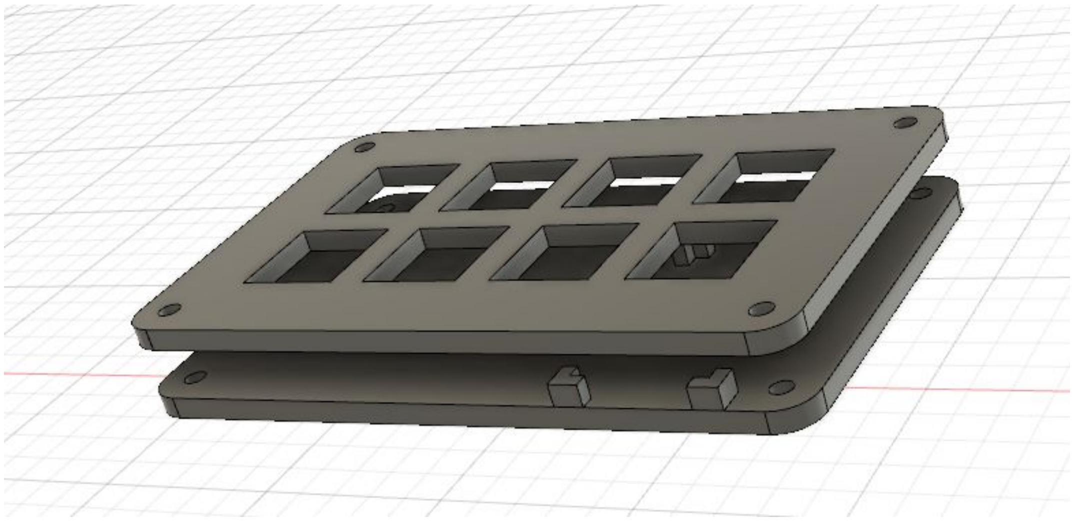

Figure <>.

A slightly different design using a Feather 32u4 employs the same sandwich design here in real life; the section for the Pro Micro is replaced with mounting holes for the Feather series of boards:

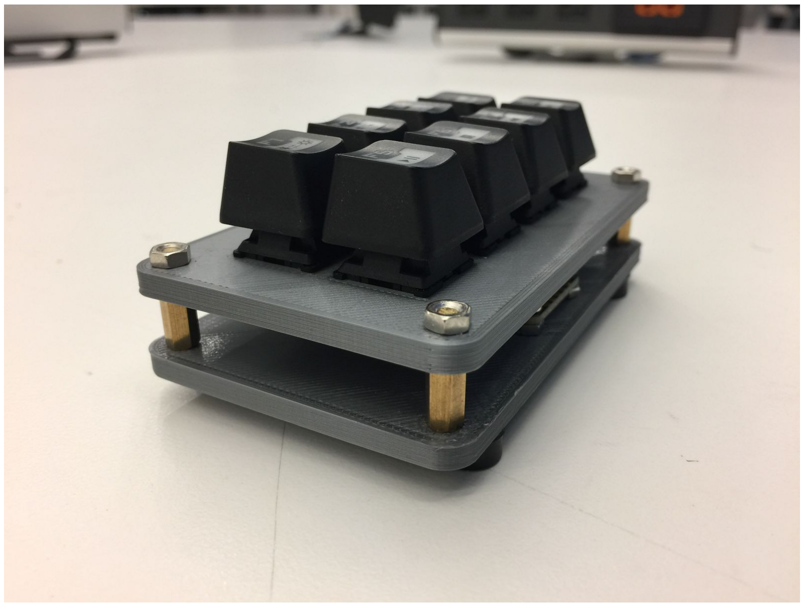

Figure <>.

Plate and Tub

Another way to complete the keyboard is by using a plate/tub design. This method requires less components, requiring only 4 screws to hold the design together.

1. **We follow the "Plate" design once more and recreate the bottom piece; only this time without the switch *and* mounting holes:**

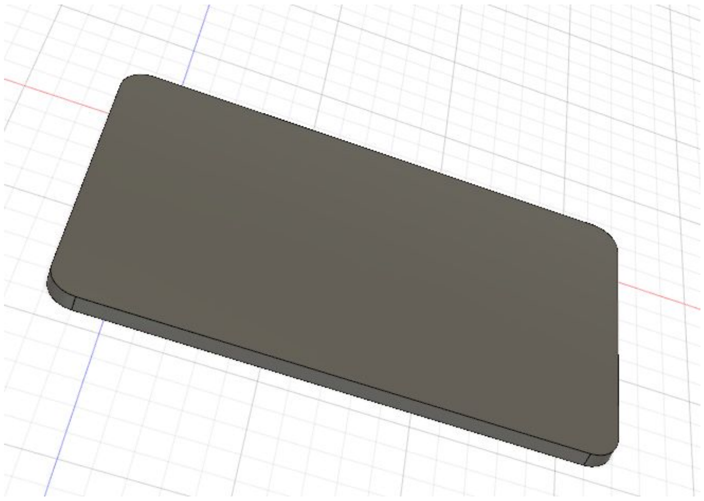

Figure <>.

2. We then create an offset (an inside perimeter within the face/shape) of 3 mm to make our inside walls:

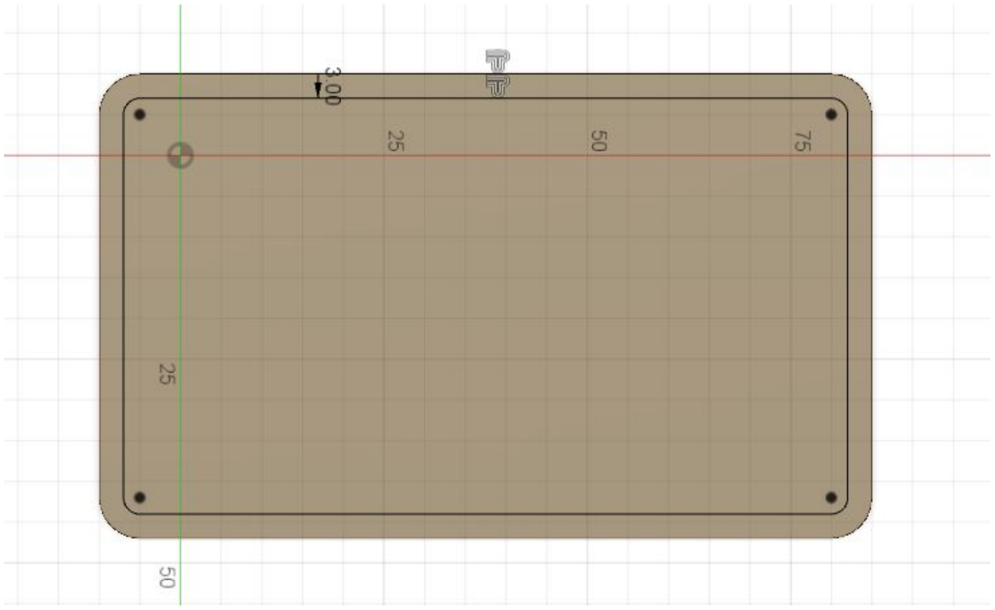

Figure <>.

3. We then extrude the offset shape by 15 mm to get our inside walls:

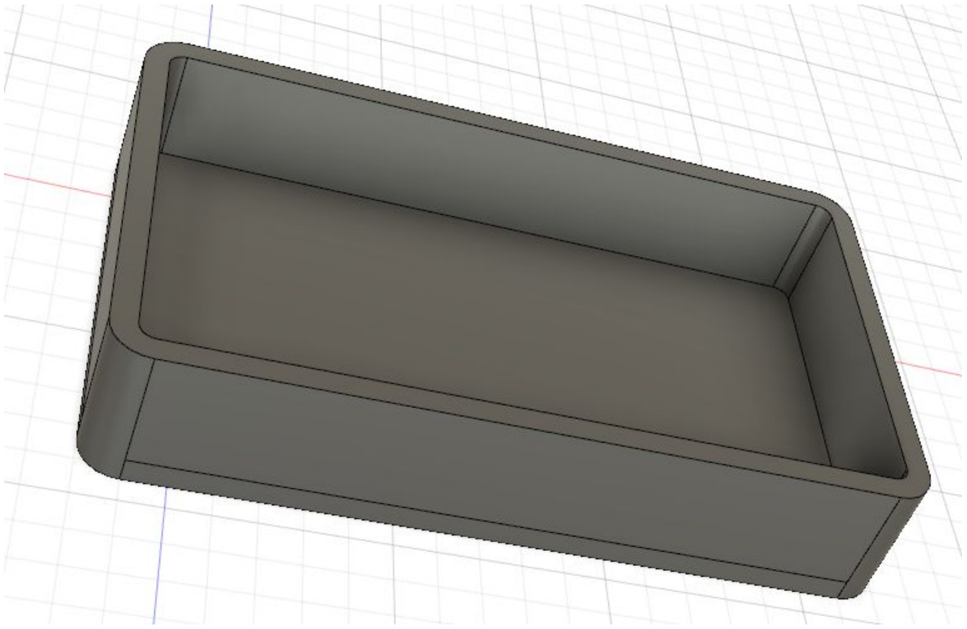

Figure <>.

4. We then add 3mm circles to the corners of the case in order to get our screw holes. Hint: we use the diameter of the rounded edge to do the measuring work for us (the center of the rounded corner as shown by the thick black dots):

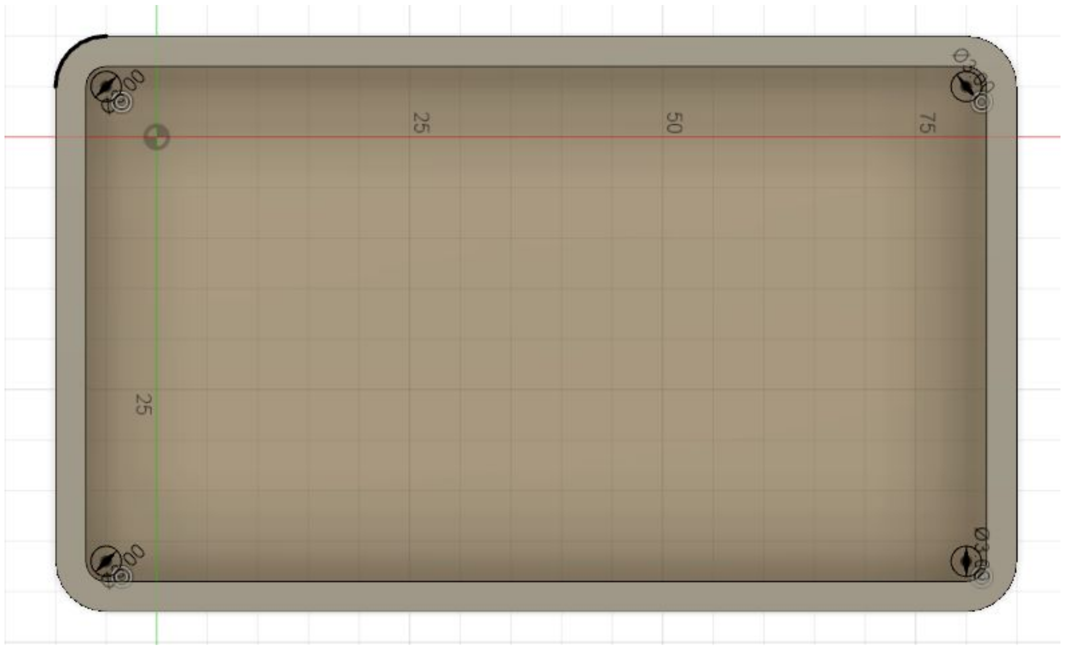

Figure <>.

5. We then use the "Center Arc" tool to create an arc around our hole and the wall of our case:

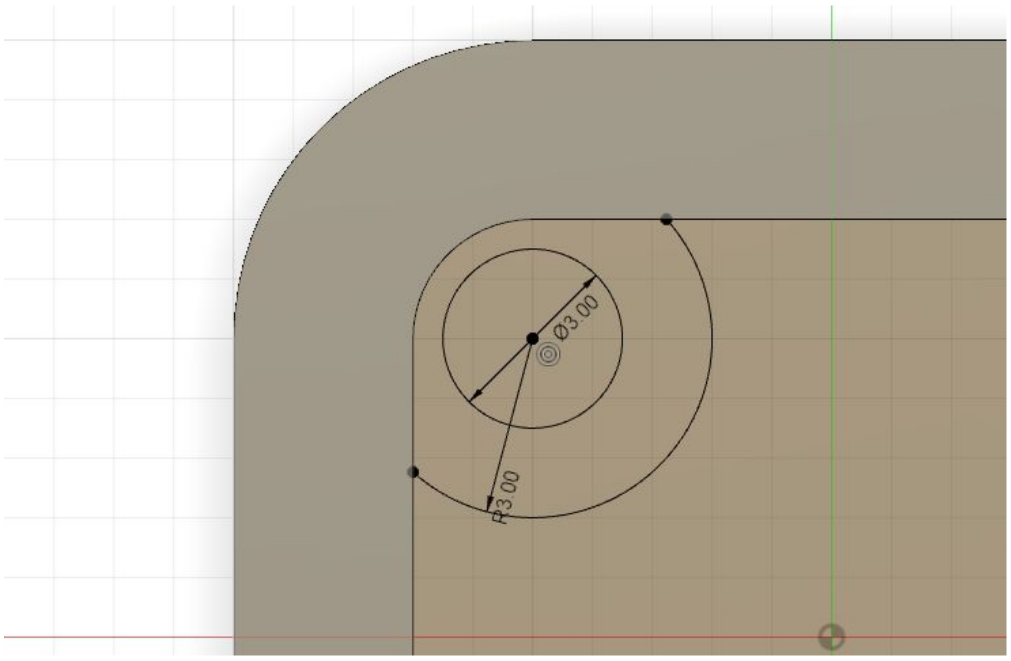

Figure <>.

6. We then extrude all four corners to get our screw holes for our case:

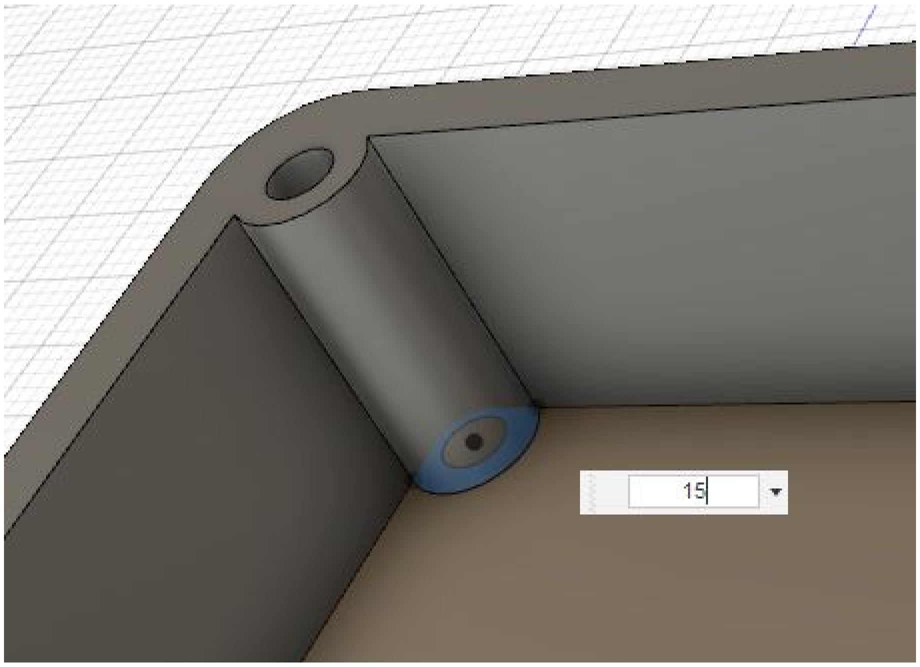

Figure <>.

It should look like this when done:

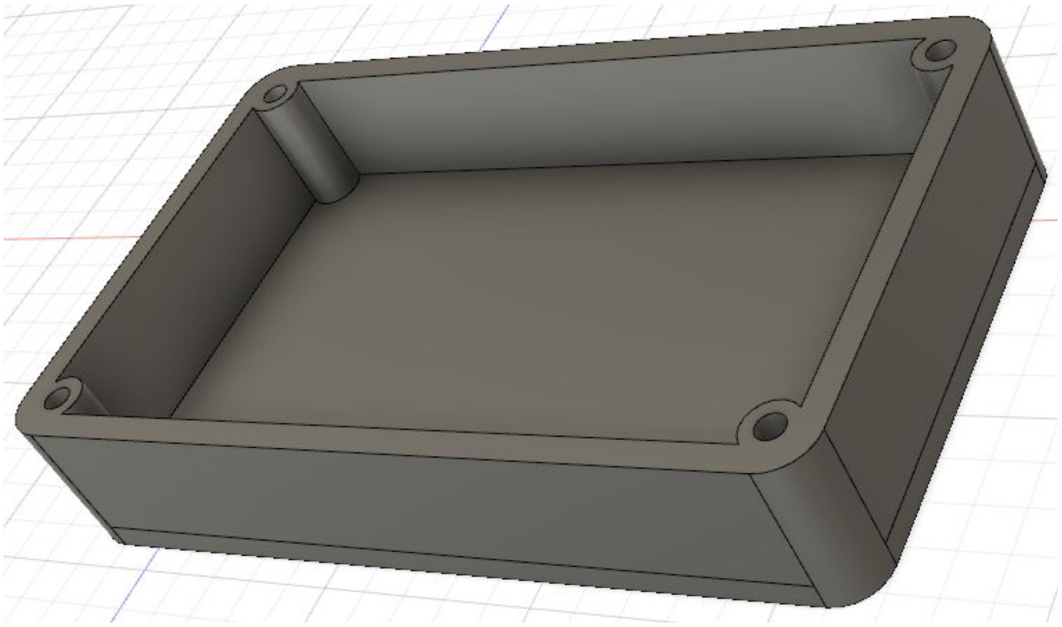

Figure <>.

At this point, like with the “Sandwich” design, we must now try to create a section to hold our Pro Micro.

7. We make a section that is slightly larger than 1.3” x 0.7” to hold our Pro Micro in place, as we might need that wiggle room. In this case, I want to center the Pro Micro as well, but you may think otherwise and place it off-center.

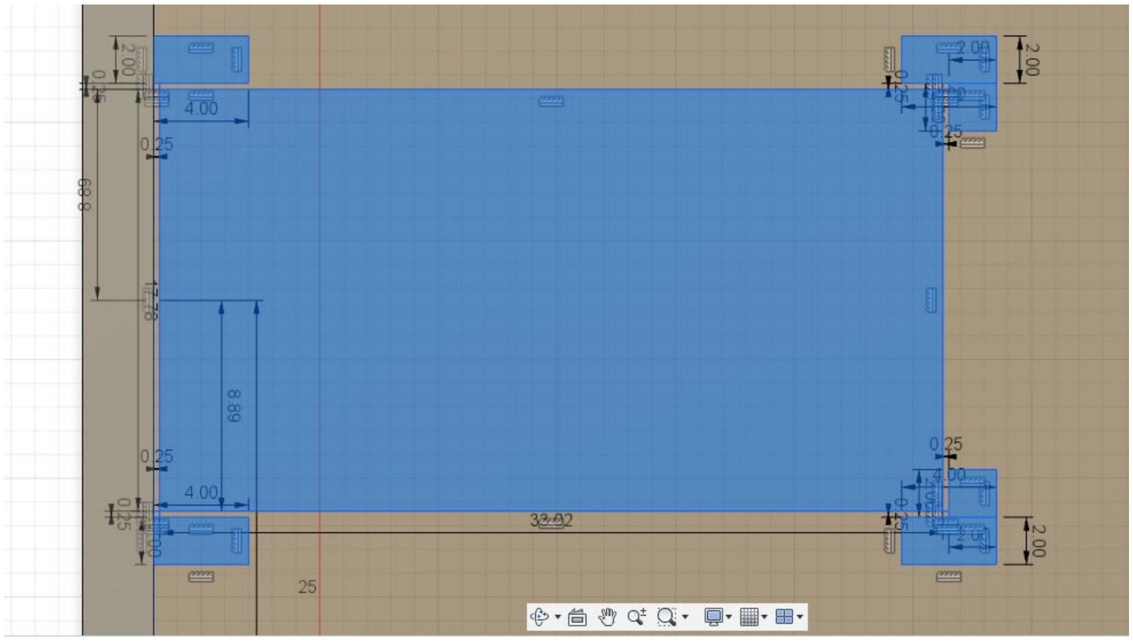

Figure <>.

In the diagram above, the four corners will hold the Pro Micro (the big rectangle in the center) in place (shown by the four L-shaped corners). Note the slight spacing that was added in order to compensate for 3D printing and slightly-off spec dimensions of the PCB (could be slightly larger or smaller).

8. We then extrude the corners by 3mm to finish the section.

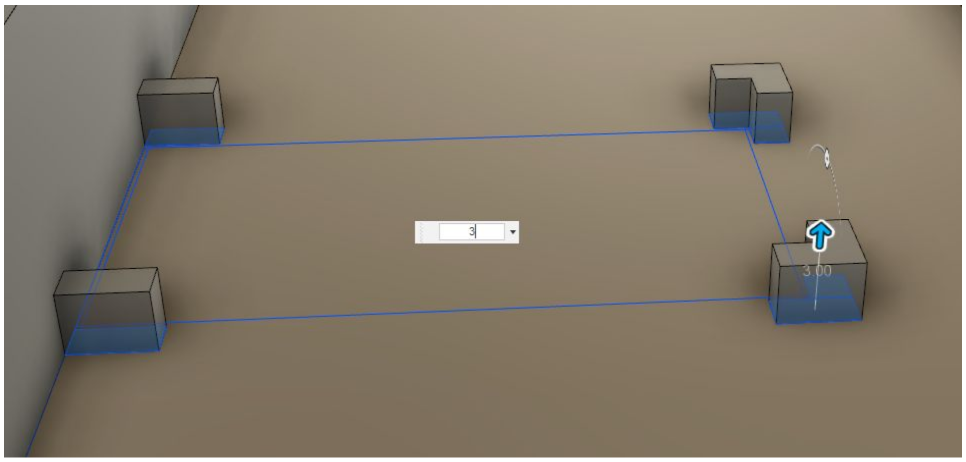

Figure <>.

Overall, it looks like this when finished:

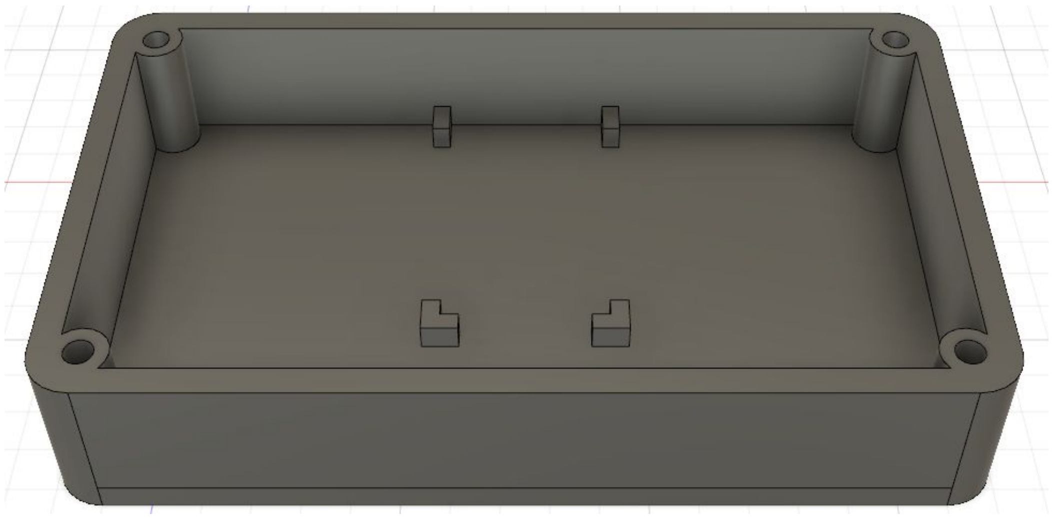

Figure <>.

We are almost done! However, we still need to be able to insert out USB cable, remember? How are we gonna do that? There's nowhere to plug the USB cable through. We need to make a *cutout* for our USB port on our Pro Micro:

9. We mark 2mm lines off the front corners of our model:

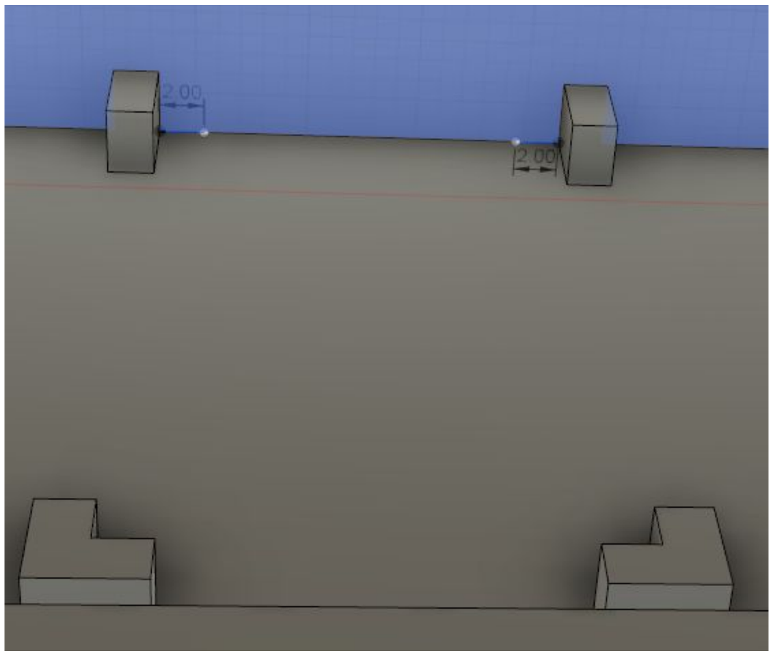

Figure <>.

10. We construct a 14.28mm x 8mm rectangle to create our cutout shape

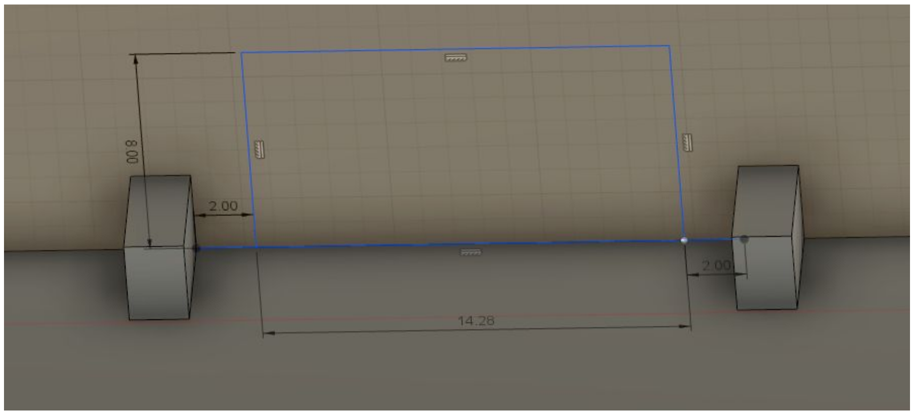

Figure <>.

11. We then use that shape to cut out our USB port:

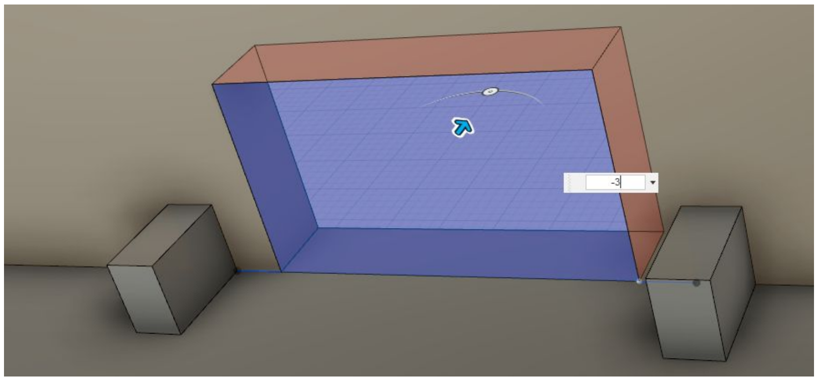

Figure <>.

The overall finished tub design looks like this:

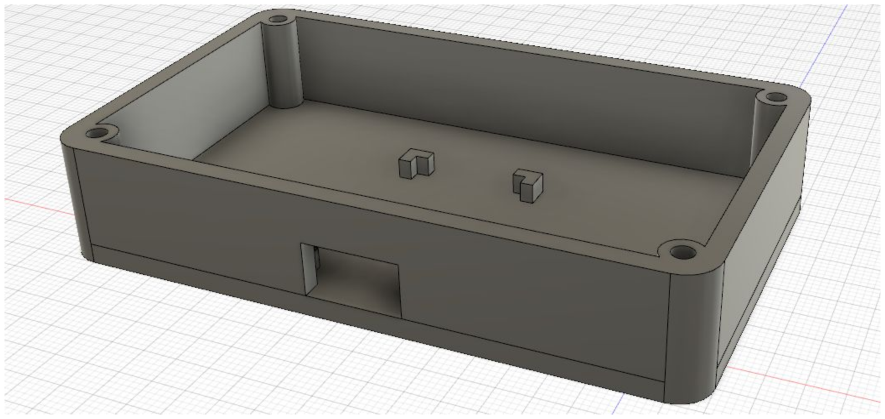

Figure <>.

Step 4: Sanity Check

1. Make sure your top plate aligns with the bottom. Make sure the holes line up as well:

Sandwich:

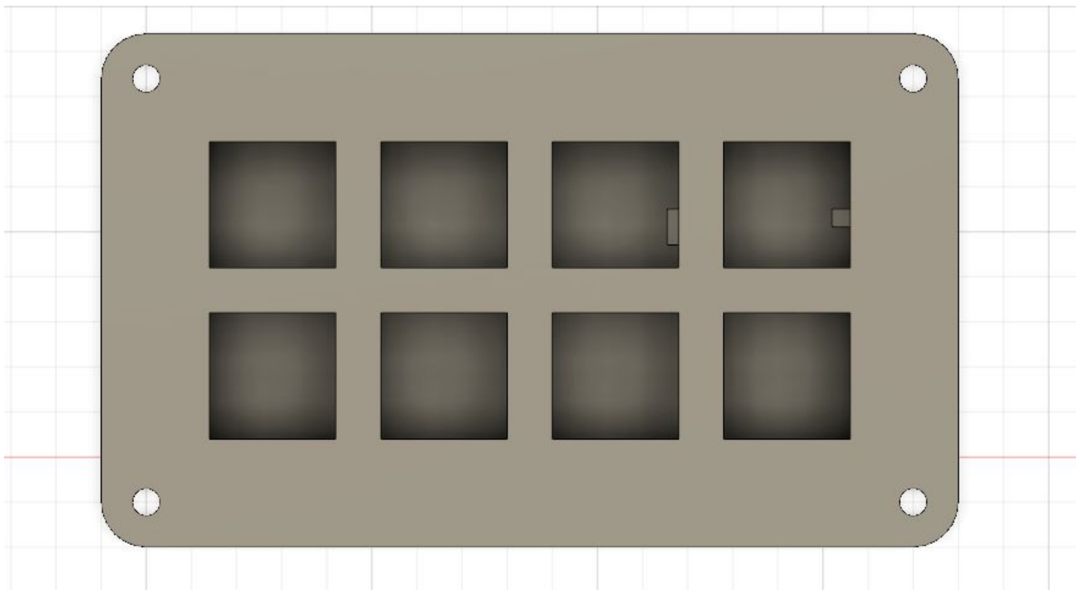

Figure <>.

Plate/Tub:

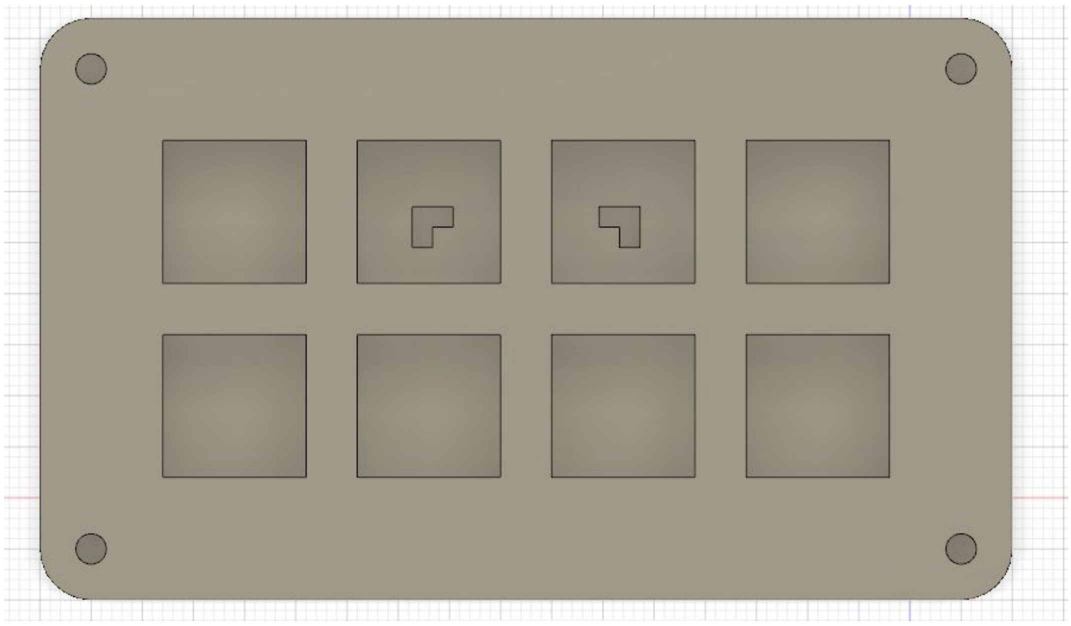

Figure <>.

To closely inspect the Plate/Tub alignment, one can check by individually zooming into the corners where the screws will mount:

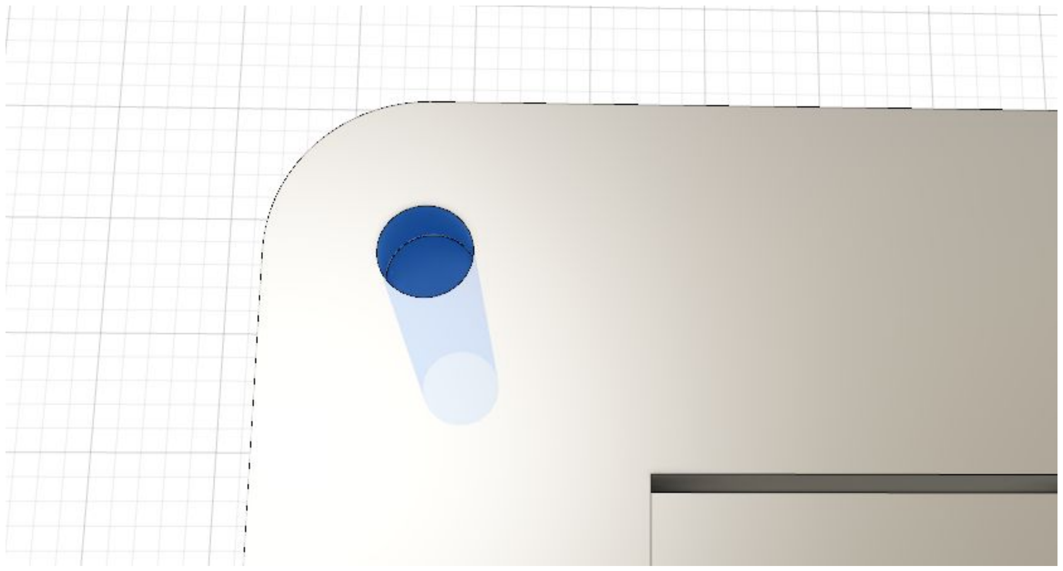

Figure <>.

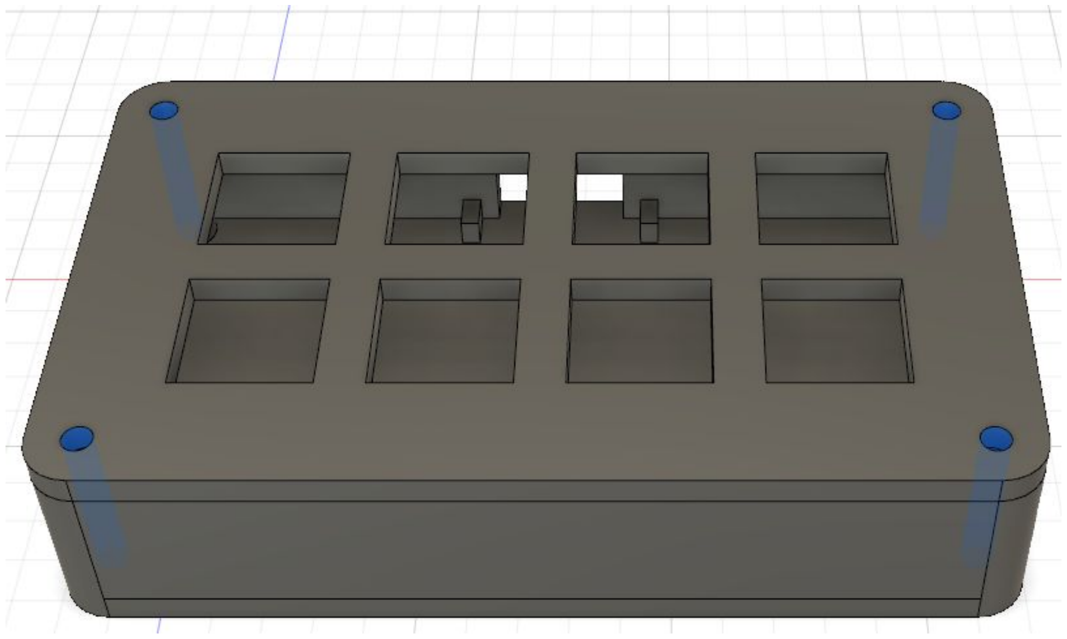

Figure <>.

2. Download 3D models of your components and try to fit them into your case.

This is your test fit in software right before you start 3D printing your case. For this to work, you must find dimensionally accurate models that will represent the components that you are fitting.

For example, Sparkfun provides a dimensionally accurate model of their Pro Micro in STL format. Insert the STL into Fusion, then orient it in such a way that it fits inside the holder:

Sandwich:

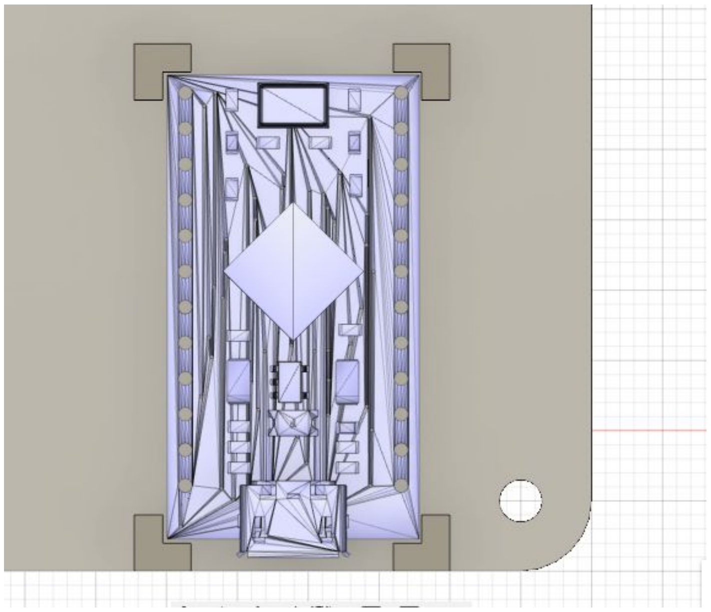

Figure <>.

The side view:

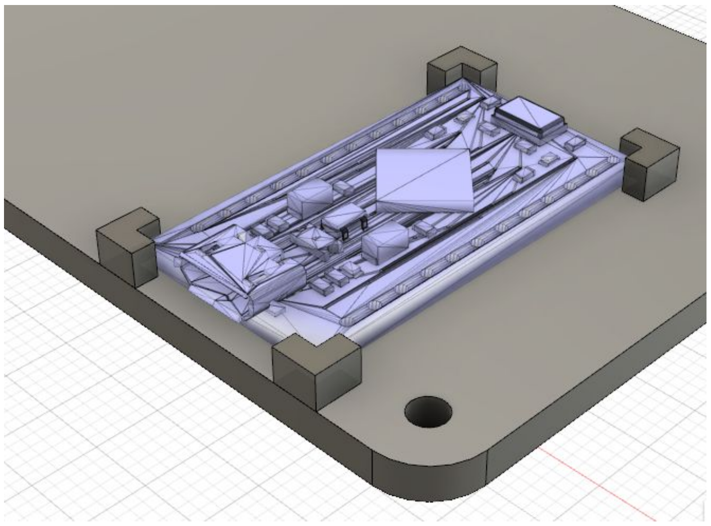

Figure <>.

Plate/Tub:

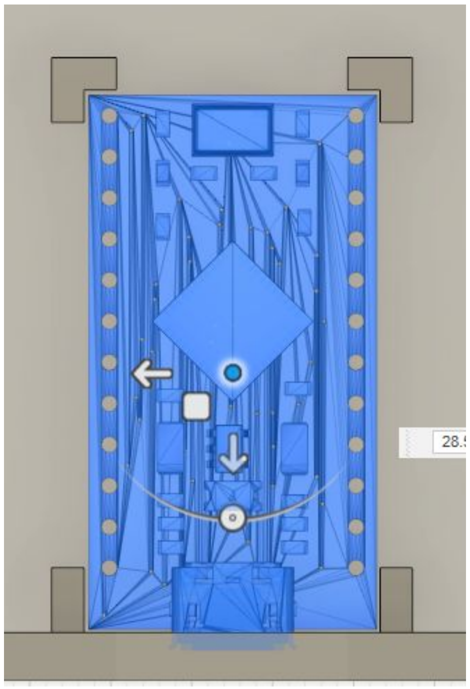

Figure <>.

Here’s a side view that shows the USB cutout:

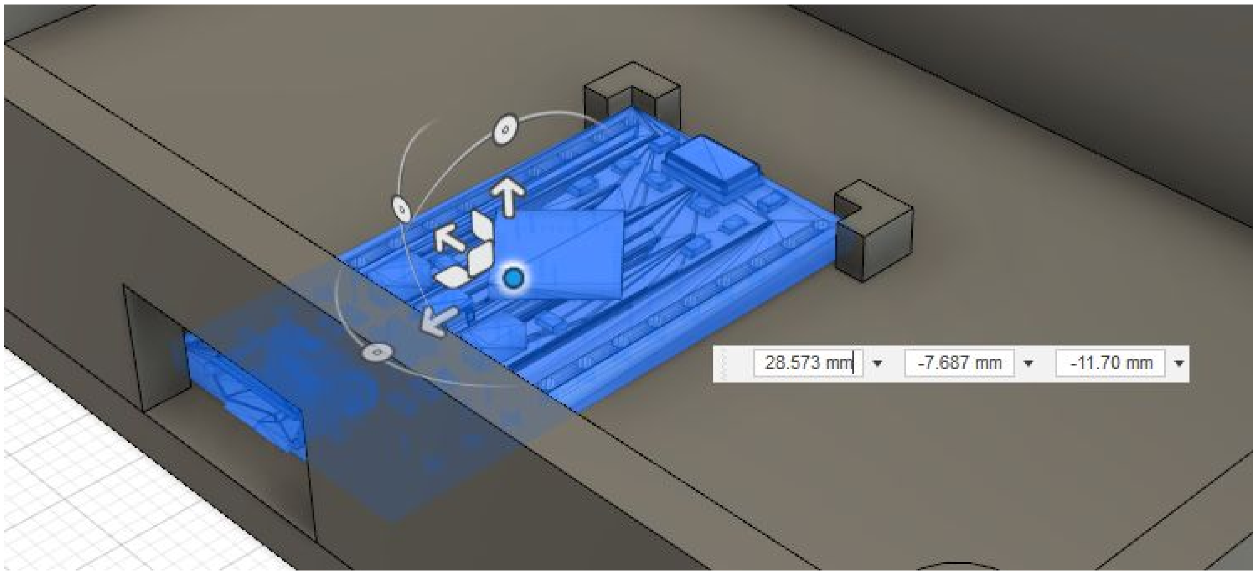

Figure <>.

Do the same if you find a Cherry MX model that is dimensionally accurate. Sites like GrabCAD and Thingiverse should be able to provide existing models for you.

Assembly

- Fit all the switches into the top plate
- Solder the components as shown in Figure <something>
    - Decide your input pins on your controller
    - Decide which switch pins is ground and which is input
    - Connect all the ground pins of your switches together
    - Connect your shared ground to the controller ground
    - Connect the inputs of your switches to the pins you chose earlier
- Position the plate/controller inside the box
- Screw the top plate down
- Voila, your keyboard is done

Credits

- Design diagrams are done in Fusion 360
- Electric diagrams are done in Fritzing