```
// note: please select all text and paste it into a .openscad file

// size of the base of the cover in mm
cover w = 75;
cover h = 75;
cover thickness = 2;

// set to 0 if not wanting buttons on top
use buttons = 0;
use resethole = 0;

// size of the sensor hole and skirt
sensor w = 34;
sensor h = 20;
sensor y = 10 + (sensor h * 0.5);

// mountingholes distansce from corners and size
mount x = 5;
mount y = 5;
mount holesize = 3 + 0.5; // 3 is too small

cover chamfer radius = 5;

// TTGO module casing
modulecase t = 1.0;
modulecase w = (cover w - 10); // + modulecase t * 2; // outer 52
modulecase h = 30; // outer
modulecase x = 5; // + (modulecase w * .5);
modulecase y = 39 - modulecase t; // + (modulecase h * .5);

modulecase z = 17.5;
modulerounding = 3.0;

// cutout in coverbase to avoid solderlugs of header connector
cover cutout h = 36; // area to be kept free
cover cutout w = 32;
cover cutout z = 1.4; // height of solder lugs
cover cutout x = (cover w/2)-(cover cutout w/2); // location of cutout
cover cutout y = modulecase y+(modulecase h/2)-(cover cutout h/2);

// display box cutout
display hole w = 26;
display hole h = 17.9;
display hole x = 18 + (modulecase t * 2); // + (displaycase w / 2);
display hole y = 6.1;
display hole z = 1.0;

// usb connector cutout
display conn w = 9;
display conn h = 4;
display conn x = 0;
display conn y = 11;
display conn z = 17.5 - modulecase t - display conn h;

// reset button cutout
reset x = (modulecase t * 2) + 10;
reset h = 5;
reset z = 17.5 - modulecase t - reset h - 1.0;
reset w = 5;

// buttons op top, size parameters
button r = 1.0;
button t = modulecase t;
button w = 7;
button h = 4;
button petal len = 10;
button petal w = 2;
button petal t = 1.0;
button s = 0.6; // space between button and case

// buttons 0 and 1 on top
button x = 6;
button z = modulecase z;
button_y0 = 5;
```

```
button y1 = 20;

// skirt
skirt t = 1.0; // thickness of wall of sensor skirt
skirt x = (cover w/2)-(sensor w/2) - skirt t;
skirt y = sensor y  - (sensor h/2) - skirt t;
skirtrounding = 3.0;
skirt w = 35;
skirt h = 20;
skirt z = 12;
skirthole w = 27;
skirthole h = 20;
skirthole ofs x = skirt t+(skirt w/2)-(skirthole w/2);
skirthole ofs y = skirt t;
skirtholerounding = 3.0;
skirt zridge = 10; // TODO

// rounder holes, use bigger number
$fn =  20;

module mountinghole()
{
    cylinder(h=cover thickness, r=mount holesize / 2);
}

// below module is a modified sample from https://danielupshaw.com/openscad-rounded-corners/
module roundedcube(size = [1, 1, 1], center = false, radius = 0.5, apply to = "all") {
    translate min = radius;
    translate xmax = size[0] - radius;
    translate ymax = size[1] - radius;
    translate zmax = size[2] - radius;

    diameter = radius * 2;

    module build point(type = "sphere", rotate = [0, 0, 0]) {
        if (type == "sphere") {
            sphere(r = radius);
        } else if (type == "cylinder") {
            rotate(a = rotate)
            cylinder(h = diameter, r = radius, center = true);
        }
    }

    obj translate = (center == false) ?
        [0, 0, 0] : [
            -(size[0] / 2),
            -(size[1] / 2),
            -(size[2] / 2)
        ];

    translate(v = obj translate) {
        hull() {
            for (translate x = [translate min, translate xmax]) {
                x at = (translate x == translate min) ? "min" : "max";
                for (translate y = [translate min, translate ymax]) {
                    y at = (translate y == translate min) ? "min" : "max";
                    for (translate z = [translate min, translate zmax]) {
                        z at = (translate z == translate min) ? "min" : "max";

                        translate(v = [translate x, translate y, translate z])
                        if (
                            (apply to == "all") ||
                            (apply to == "xmin" && x at == "min") || (apply to == "xmax"    ↵
                            && x at == "max") ||
                            (apply to == "ymin" && y at == "min") || (apply to == "ymax"    ↵
                            && y at == "max") ||
                            (apply to == "zmin" && z at == "min") || (apply to == "zmax"    ↵
                            && z at == "max")
                        ) {
                            build point("sphere");
                        } else {
                            rotate =
                                (apply to == "xmin" || apply to == "xmax" || apply to ==    ↵
                                "x") ? [0, 90, 0] : (
```

```
                                (apply to == "ymin" || apply to == "ymax" || apply to ==
                                "y") ? [90, 90, 0] :
                                [0, 0, 0]
                            );
                            build point("cylinder", rotate);
                        }
                    }
                }
            }
        }
    }
}

module chamfer outside()
{
    t = cover thickness;
    r = cover chamfer radius;
    difference() {
        cube([r, r, t]);
        difference() {
            cube([r, r, t]);
            difference() {
                cube([r*2, r*2, t]);
                translate([r, r, 0]) {
                    cylinder(h=t, r=r);
                }
            }
        }
    }
}

module chamfers outside()
{
    translate([0,0,0]) { // top left
        rotate([0,0,0]) {
            chamfer outside();
        }
    }
    translate([0,cover h,0]) { // top right
        rotate([0,0,-90]) {
            chamfer outside();
        }
    }
    translate([cover w,cover h,0]) { // top right
        rotate([0,0,-180]) {
            chamfer outside();
        }
    }
    translate([cover w,0,0]) { // top right
        rotate([0,0,90]) {
            chamfer outside();
        }
    }
}

module mountingholes()
{
    translate([0,0,0]) { // top left
        translate([mount x, mount y, 0]) {
            mountinghole();
        }
    }
    translate([cover w,0,0]) { // top right
        translate([-mount x, mount y, 0]) {
            mountinghole();
        }
    }
    translate([cover w,cover h,0]) { // bottom right
        translate([-mount x, -mount y, 0]) {
            mountinghole();
        }
    }
    translate([0,cover h,0]) { // bottom left
        translate([mount_x, -mount_y, 0]) {
```

```
            mountinghole();
        }
    }
}

module button add() {
    union() {
        roundedcube(size=[button w,button h,button t],
            radius = button r, apply to = "zmax");
        translate([0, button petal w/2, 0]) {
            cube([button petal len, button petal w, button petal t]);
        }
    }
}

module button subtract()
{
    difference() {
        union() {
            translate([-button s, -button s, 0]) {
                roundedcube(size=[button w+button s,button h+2*button s,button t],
                    radius = button r, apply to = "z");
            }
            translate([0, (button petal w/2)-button s, 0]) {
                cube([button petal len, button petal w+2*button s, button petal t]);
            }
        }
        translate([0,0,0]) {
            button add();
        }
    }
}

module buttons add()
{
    translate([button x, button y0, button z-modulecase t]) {
        button add();
    }
    translate([button x, button y1, button z-modulecase t]) {
        button add();
    }
}

module buttons subtract()
{
    translate([button x, button y0, button z-modulecase t]) {
        button subtract();
    }
    translate([button x, button y1, button z-modulecase t]) {
        button subtract();
    }
}

module buttons support()
{
    buttonsupport x = button x + 7;
    buttonsupport y = 2*modulecase t;
    buttonsupport w = modulecase w - buttonsupport x - modulecase t - 0.5; // 1;
    buttonsupport h = modulecase h - 4*modulecase t;
    buttonsupport z = 1.5;
    translate([buttonsupport x, buttonsupport y, button z-2*modulecase t]) {
        cube([buttonsupport w, buttonsupport h, buttonsupport z]);
    }
}

module reset button()
{
   translate([reset x, 0, reset z ]) {
     cube([reset w , 2*modulecase t, reset h]); //
   }
}

module module displayhole()
{
```

```
   // dup below
   translate([display hole x, display hole y, modulecase z]) {
        translate([0,0, -2*modulecase t]) {
            cube([display hole w, display hole h, 2*modulecase t]);
        }
   }
}

module ttgomodulecase()
{
    translate([modulecase x, modulecase y, 0]) {
       union() {
          // extra frame around display
           if (0) {
               display hole ridge t = 1.0;
               translate([display hole x, // display hole x-display hole ridge t
                          display hole y, // display hole y-display hole ridge t
                            modulecase z-modulecase t-display hole z]) { //
                    difference() {
                        cube([display hole w, // display hole w+2*display hole ridge t
                              modulecase h, // display hole h+2*display hole ridge t
                              display hole z]);
                        translate([display hole ridge t,display hole ridge t,0]) {
                            cube([display hole w,
                                  display hole h, display hole z]);
                        }
                    }
               }
           }

           difference() {
               // solid outside
               union() {
                   roundedcube(size = [modulecase w, modulecase h, modulecase z],
                        radius = modulerounding, apply to = "zmax");
                   if (use buttons) {
                       // buttons on top
                       buttons add();
                   }
               }
               if (use buttons) {
                   // void around buttons
                   buttons subtract();
               }

               // inner void
               translate([modulecase t, modulecase t, 0]) {
                    roundedcube(size = [
                        modulecase w-2*modulecase t,
                        modulecase h-2*modulecase t,
                        modulecase z - modulecase t], radius = modulerounding, apply to =
                        "zmax");
               }
               // more void inside top for TTGO module to fit snugly against connector side
               translate([modulecase t, modulecase t, modulecase z-2*modulecase t]) {
    //              cube([modulecase w-2*modulecase t, modulecase h-2*modulecase t,
    modulecase t]);
               }

               // hole for display
               module displayhole();

               // hole for connector on right side
               translate([0, display conn y, display conn z- modulecase t ]) {
                 cube([2*modulecase t , display conn w,display conn h ]); //
               }
               if (use resethole) {
                   // hole for reset button
                   reset button();
               }
           }

           difference() {
               // extra support for button hinges
```

```
                buttons support();

                // hole for display
                module displayhole();
            }

        }
    }
}

module baseplate()
{
    difference() {
        union() {
            difference() {
                // base covering full PCB
                cube([cover w,cover h,cover thickness]);

                // subtractive parts below

                // rounded chamfers on edges of cover
                chamfers outside();

                // sensor cutout
                translate([cover w/2,sensor y,0]) {
                    translate([-sensor w/2,-sensor h/2,-cover thickness]) {
                        roundedcube(size=[sensor w,sensor h,cover thickness],
                            radius = modulerounding, apply to = "zmax");
                    }
                }
                // subtract mounting holes as well
                mountingholes();
                // subtract displaycase hole
                translate([modulecase x,modulecase y,0]) {
                    roundedcube(size=[modulecase w,modulecase h,cover thickness],
                        radius = modulerounding, apply to = "zmax");
                }
            }
            // drop TTGO modulecase in hole (again)
            ttgomodulecase();
        }
        // cutout in cover to avoid solder
        translate([cover cutout x, cover cutout y, 0]) {
            cube([cover cutout w, cover cutout h, cover cutout z]);
        }
        // another cutout to prevent solder
        translate([cover w/2,sensor y,0]) {
            sensor cutout w = sensor w + 8.0;
            sensor cutout h = 14;
            translate([-sensor cutout w/2,
                       -sensor cutout h/2,
                       0]) {
                cube([sensor cutout w,
                      sensor cutout h,
                      cover cutout z]);
            }
        }
    }

}

module sensorskirt()
{
    translate([skirt x,skirt y,0]) {
        difference() {
            roundedcube(size=[skirt w+2*skirt t, skirt h+2*skirt t, skirt z+2*skirt t],
                radius = skirtrounding, apply to = "zmax");
            // subtract inside
            translate([skirt t,skirt t,0]) {
                roundedcube(size=[skirt w,skirt h,skirt z+skirt t],
                    radius = skirtrounding, apply to = "zmax");
            }
            // subtract hole in top
            translate([skirthole_ofs_x, skirthole_ofs_y, -skirt_t]) {
```

```
                roundedcube(size=[27,20,skirt z*4],
                    radius = skirtholerounding, apply to = "all");
            }
        }
    }
}


baseplate();

translate([0,68,0]) {
    sensorskirt();
}
```