Supplementary Material for ChatBuilder: LLM-assisted Modular Robot Realization

Xin Chen[1], Zherong Pan[2], Xifeng Gao[2], Aiguo Song[1] and Lifeng Zhu[1,*]

I. MODULE LIBRARY

We use OpenSCAD programming modeling module library. There are four different types of parts in the basic module library, namely Joint, Link, Chassis and Wheel. We also include supplementary modules such as the Leg, Leg-wheel, and Mecanum-wheel.

A. Predefined Geometric Function

Before creating these modules, we first write some editable geometry functions. Calling these functions can generate hexagon prisms, triangular prisms, cubes, and cylinders. By doing so, we simplify the code and ensure reusability for different geometric shapes. The OpenSCAD code is as follows:

a) : Triangular Prism.

```
//File name: triangular_prism_f.scad
//Purpose: generate a triangular prism.
//Parameter:
//1. fillet: the fillet radius.
//2. height: the height of the triangular prism.
//3. circum_radius: the circumradius of the base.
module triangular_prism(height, circum_radius) {
    side_length = circum_radius * sqrt(3);
    points = [
        [0, -circum_radius],
        [side_length / 2, circum_radius * sin(60)],
        [-side_length / 2, circum_radius * sin(60)]
    ];
    linear_extrude(height = height)
    polygon(points);
}
module triangular_prism_f(fillet=5, height, circum_radius){
    rotate([0,180,0])
    translate([0,0,circum_radius * sin(60)+fillet/2])
    minkowski(){
    rotate([-90,0,0])
    translate([0,0,-(height-2*fillet)/2])
    triangular_prism(height-2*fillet, circum_radius-fillet);
    sphere(fillet);
    }
 }
```

b) : Hexagon Prism.

```
//File name: hexagon_prism_f.scad
//Purpose: generate a hexagon prism.
//Parameter:
//1. fillet: the fillet radius.
//2. h: the height of the hexagon prism.
//3. r: the circumradius of the base.
module hexagon_prism(height=10, radius=5) {
    side_length = radius * cos(30);
    linear_extrude(height=height)
        polygon(points=[
            [radius * cos(0), radius * sin(0)],
            [radius * cos(60), radius * sin(60)],
            [radius * cos(120), radius * sin(120)],
            [radius * cos(180), radius * sin(180)],
```

*Corresponding author, lfzhulf@gmail.com

[1]Xin Chen, Aiguo Song and Lifeng Zhu are with the State Key Laboratory of Digital Medical Engineering, Jiangsu Key Lab of Robot Sensing and Control, School of Instrument Science and Engineering, Southeast University, China

[2]Zherong Pan and Xifeng Gao are with the LightSpeed Studios Seattle, WA, USA

```
            [radius * cos(240), radius * sin(240)],
            [radius * cos(300), radius * sin(300)]
        ]);
}
module hexagon_prism_f(fillet=2,h,r){
    minkowski(){
    translate([0,0,fillet])
    hexagon_prism(h-2*fillet,r-fillet);
    sphere(fillet);
    }
}
```

c) : Cube with Fillet.

```
//File name: cube_f.scad
//Purpose: generate a cube with fillet.
//Parameter:
//1. fillet: the fillet radius.
//2. w: the length of the cube.
//3. d: the width of the cube.
//4. h: the height of the cube.
module cube_f(fillet=2,w,d,h){
    minkowski(){
    translate([fillet,fillet,fillet])
    cube([w-2*fillet,d-2*fillet,h-2*fillet]);
    sphere(fillet);
    }
}
```

d) : Cylinder with Fillet.

```
//File name: cylinder_f.scad
//Purpose: generate a cylinder with fillet.
//Parameter:
//1. fillet: the fillet radius.
//2. height: the height of the cylinder.
//3. rad1: the bottom radius of the cylinder.
//4. rad2: the top radius of the cylinder.
module cylinder_f(fillet=2,height,rad1,rad2){
    minkowski(){
    translate([0,0,fillet])
    cylinder(h=height-2*fillet,r1=rad1-fillet,r2=rad2-fillet);
    sphere(fillet);
    }
}
```

e) : Male Connector.

```
//File name: male_connector.scad
//Purpose: generate a male connector.
//Parameter:
//1. fillet: the fillet radius.
//2. connection_h: the length of the connector.
//3. connection_r: the circumradius of the connector.
//4. screw_r: the radius of the screw hole.
include<hexagon_prism_f.scad>
module male_connector(fillet=2,connection_h=20,connection_r=25,screw_r=3){
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
```

f) : Female Connector.

```
//File name: female_connector.scad
//Purpose: generate a female connector.
//Parameter:
//1. fillet: the fillet radius.
//2. connection_h: the length of the connector.
//3. connection_r: the circumradius of the connector.
//4. thickness: wall thickness.
//5. screw_r: the radius of the screw hole.
include<hexagon_prism_f.scad>
module female_connector(fillet=2,connection_h=20,connection_r=25,thickness=8,screw_r=3){
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
    translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
        )])
    cube(2*(connection_r+thickness));
    translate([0,0,thickness])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([0,0,thickness+connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r+thickness)])
    cylinder(4*(connection_r+thickness),screw_r,screw_r);
    }
}
```

B. Basic Modules

Before writing the files for basic modules such as Joint, Link, Chassis, and Wheel, include the above-mentioned geometries. Use these geometries along with OpenSCAD's built-in functions to create the basic module library (the tread pattern of the tire in the Wheel component is sourced from tire-v3.stl). The OpenSCAD code is as follows:

a) : Joint. The motor stand and motor coupling are collectively referred to as the Joint.

```
//File name: motor_stand_f.scad
//Purpose: generate a motor stand.
//Parameter:
//1. display: display settings. when set to true, display the entire assembled module; when
    set to a specific number, the corresponding number of parts is displayed.
//2. cube_shape: joint shape settings. set to true for a cube shape, and set to false for a
    cylinder shape.
//3. motor_axis_offset_r: the radius of the motor mount hole.
//4. fillet: the fillet radius.
//5. connection_h: the length of the connector.
//6. connection_r: the circumradius of the connector.
//7. screw_r: the radius of the screw hole.
//8. motor_width: the width of the motor.
//9. motor_height: the height of the motor.
//10. motor_screw_hole_position: the shortest distance from the center of the screw hole to
    the edge.
//11. motor_axis_offset_h: the height of the motor mount hole.
//12. motor_axis_length: the length of the motor shaft.
//13. thickness: wall thickness.
include<hexagon_prism_f.scad>
include<cylinder_f.scad>
include<cube_f.scad>
include<triangular_prism_f.scad>
module motor_stand_cube_f(display=true,motor_axis_offset_r=35,fillet=2,connection_h=20,
    connection_r=25,screw_r=3,motor_width=80,motor_height=123,motor_screw_hole_position=8.2,
    motor_axis_offset_h=3,motor_axis_length=32,thickness=8){
if  (display==true){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2 ;
color([0.5,0.5,0.5])
union(){
    translate([0,0,-prolonging/2])
    difference(){
        translate([0,0,-motor_height/2])
        translate([(-motor_width-2*thickness-(screw_r+3)*4)/2,-(motor_width+2*thickness+(
            screw_r+3)*4)/2,0])
```

```
    cube_f(5,(screw_r+3)*4+motor_width+2*thickness,(screw_r+3)*4+motor_width+2*thickness,
        motor_height+thickness+prolonging);
    translate([0,0,-(motor_height+thickness*2)/2-thickness])
    translate([-(motor_width+(screw_r+3)*4)/2,-(motor_width+(screw_r+3)*4)/2,0])
    cube_f(5,(screw_r+3)*4+motor_width,(screw_r+3)*4+motor_width,motor_height+thickness
        *2);
    translate([0,0,0])
    cylinder(motor_height+thickness,motor_axis_offset_r+2,motor_axis_offset_r+2);
    rotate([0,0,-45])
    translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
    cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
    rotate([0,0,45])
    translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
    cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
    translate([0,0,(motor_height+prolonging+2*thickness)/2-connection_r/2])
    translate([0,0,prolonging/2])
    translate([-2*screw_r,-sqrt(2)*motor_width,-2*screw_r])
    cube_f(4,4*screw_r,sqrt(2)*motor_width*2,100*screw_r+thickness);
    translate([motor_width/2-motor_screw_hole_position,motor_width/2-
        motor_screw_hole_position,0])
    cylinder(motor_height+thickness,screw_r,screw_r);
    translate([motor_width/2-motor_screw_hole_position,-(motor_width/2-
        motor_screw_hole_position),0])
    cylinder(motor_height+thickness,screw_r,screw_r);
    translate([-(motor_width/2-motor_screw_hole_position),motor_width/2-
        motor_screw_hole_position,0])
    cylinder(motor_height+thickness,screw_r,screw_r);
    translate([-(motor_width/2-motor_screw_hole_position),-(motor_width/2-
        motor_screw_hole_position),0])
    cylinder(motor_height+thickness,screw_r,screw_r);
    translate([0,0,prolonging/2])
    triangular_prism_f(10,motor_width*4,motor_height/4);
}
translate([0,0,-prolonging/2])
translate([(motor_width+4*(screw_r+3))/2-(screw_r+3)+3,0,-motor_height/2])
difference(){
    cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
    translate([0,0,-fillet])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
}
translate([0,0,-prolonging/2])
translate([-(motor_width+4*(screw_r+3))/2+(screw_r+3)-3,0,-motor_height/2])
difference(){
    cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
    translate([0,0,-fillet])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
}
rotate([0,90,0])
translate([0,0,(screw_r+3)*2+motor_width/2+thickness])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([-(connection_r),-(connection_r),-2*(connection_r)])
cube(2*(connection_r));
translate([0,0,connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r)])
cylinder(4*(connection_r),screw_r,screw_r);
}
rotate([0,-90,0])
translate([0,0,(screw_r+3)*2+motor_width/2+thickness])
difference(){
translate([0,0,-fillet-thickness])
hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
cube(2*(connection_r));
translate([0,0,connection_h/2])
rotate([90,0,0])
```

```
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
color([0.5,0.5,0.5])
translate([0,0,-prolonging/2])
union(){
    difference(){
    translate([0,0,-thickness-motor_height/2])
    translate([-((screw_r+3)*4+motor_width+2*thickness)/2,-((screw_r+3)*4+motor_width+2*
        thickness)/2,0])
    cube_f(2,(screw_r+3)*4+motor_width+2*thickness,(screw_r+3)*4+motor_width+2*thickness,
        thickness);
    translate([(motor_width+4*(screw_r+3))/2-(screw_r+3)+3,0,-(motor_height+thickness+fillet
        *2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([(motor_width+4*(screw_r+3))/2-(screw_r+3)+3,0,prolonging/2-2-(motor_height+2*
        thickness+prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    translate([-(motor_width+4*(screw_r+3))/2+(screw_r+3)-3,0,-(motor_height+thickness+fillet
        *2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([-(motor_width+4*(screw_r+3))/2+(screw_r+3)-3,0,prolonging/2-2-(motor_height+2*
        thickness+prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    }
    rotate([180,0,0])
    translate([0,0,motor_height/2+thickness])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
}

if  (display==1){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2 ;
color([0.5,0.5,0.5])
union(){
    translate([0,0,-prolonging/2])
    difference(){
        translate([0,0,-motor_height/2])
        translate([(-motor_width-2*thickness-(screw_r+3)*4)/2,-(motor_width+2*thickness+(
            screw_r+3)*4)/2,0])
        cube_f(5,(screw_r+3)*4+motor_width+2*thickness,(screw_r+3)*4+motor_width+2*thickness,
            motor_height+thickness+prolonging);
        translate([0,0,-(motor_height+thickness*2)/2-thickness])
        translate([-(motor_width+(screw_r+3)*4)/2,-(motor_width+(screw_r+3)*4)/2,0])
        cube_f(5,(screw_r+3)*4+motor_width,(screw_r+3)*4+motor_width,motor_height+thickness
            *2);
        translate([0,0,0])
        cylinder(motor_height+thickness,motor_axis_offset_r+2,motor_axis_offset_r+2);
        rotate([0,0,-45])
        translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
        cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
        rotate([0,0,45])
        translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
        cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
        translate([0,0,(motor_height+prolonging+2*thickness)/2-connection_r/2])
        translate([0,0,prolonging/2])
        translate([-2*screw_r,-sqrt(2)*motor_width,-2*screw_r])
        cube_f(4,4*screw_r,sqrt(2)*motor_width*2,100*screw_r+thickness);
```

```
        translate([motor_width/2-motor_screw_hole_position,motor_width/2-
            motor_screw_hole_position,0])
        cylinder(motor_height+thickness,screw_r,screw_r);
        translate([motor_width/2-motor_screw_hole_position,-(motor_width/2-
            motor_screw_hole_position),0])
        cylinder(motor_height+thickness,screw_r,screw_r);
        translate([-(motor_width/2-motor_screw_hole_position),motor_width/2-
            motor_screw_hole_position,0])
        cylinder(motor_height+thickness,screw_r,screw_r);
        translate([-(motor_width/2-motor_screw_hole_position),-(motor_width/2-
            motor_screw_hole_position),0])
        cylinder(motor_height+thickness,screw_r,screw_r);
        translate([0,0,prolonging/2])
        triangular_prism_f(10,motor_width*4,motor_height/4);
    }
    translate([0,0,-prolonging/2])
    translate([(motor_width+4*(screw_r+3))/2-(screw_r+3)+3,0,-motor_height/2])
    difference(){
        cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
        translate([0,0,-fillet])
        cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    }
    translate([0,0,-prolonging/2])
    translate([-(motor_width+4*(screw_r+3))/2+(screw_r+3)-3,0,-motor_height/2])
    difference(){
        cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
        translate([0,0,-fillet])
        cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    }
    rotate([0,90,0])
    translate([0,0,(screw_r+3)*2+motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
    rotate([0,-90,0])
    translate([0,0,(screw_r+3)*2+motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
}

if(display==2){
color([0.5,0.5,0.5])
translate([0,0,-prolonging/2])
union(){
    difference(){
    translate([0,0,-thickness-motor_height/2])
    translate([-((screw_r+3)*4+motor_width+2*thickness)/2,-((screw_r+3)*4+motor_width+2*
        thickness)/2,0])
    cube_f(2,(screw_r+3)*4+motor_width+2*thickness,(screw_r+3)*4+motor_width+2*thickness,
        thickness);
    translate([(motor_width+4*(screw_r+3))/2-(screw_r+3)+3,0,-(motor_height+thickness+fillet
        *2)])
```

```
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([(motor_width+4*(screw_r+3))/2-(screw_r+3)+3,0,prolonging/2-2-(motor_height+2*
        thickness+prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    translate([-(motor_width+4*(screw_r+3))/2+(screw_r+3)-3,0,-(motor_height+thickness+fillet
        *2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([-(motor_width+4*(screw_r+3))/2+(screw_r+3)-3,0,prolonging/2-2-(motor_height+2*
        thickness+prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    }
    rotate([180,0,0])
    translate([0,0,motor_height/2+thickness])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }  }
    }
}

module motor_stand_cylinder_f(display=true,motor_axis_offset_r=35,fillet=2,connection_h=20,
    connection_r=25,screw_r=3,motor_width=80,motor_height=123,motor_screw_hole_position=8.2,
    motor_axis_offset_h=3,motor_axis_length=32,thickness=8){
    if(display==true){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
color([0.5,0.5,0.5])
union(){
    translate([0,0,-prolonging/2])
    difference(){
        translate([0,0,-motor_height/2])
        cylinder_f(5,motor_height+thickness+prolonging,sqrt(2)*motor_width/2+thickness,sqrt
            (2)*motor_width/2+thickness);
        translate([0,0,-(motor_height+thickness*2)/2-thickness])
        cylinder_f(5,motor_height+thickness*2,sqrt(2)*motor_width/2,sqrt(2)*motor_width/2);
        translate([0,0,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,motor_axis_offset_r+2,
            motor_axis_offset_r+2);
        rotate([0,0,-45])
        translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
        cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
        rotate([0,0,45])
        translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
        cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
        translate([0,0,(motor_height+prolonging+2*thickness)/2-connection_r/2])
        translate([0,0,prolonging/2])
        translate([-2*screw_r,-sqrt(2)*motor_width,-2*screw_r])
        cube_f(4,4*screw_r,sqrt(2)*motor_width*2,100*screw_r+thickness);
        translate([motor_width/2-motor_screw_hole_position,motor_width/2-
            motor_screw_hole_position,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([motor_width/2-motor_screw_hole_position,-(motor_width/2-
            motor_screw_hole_position),0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([-(motor_width/2-motor_screw_hole_position),motor_width/2-
            motor_screw_hole_position,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([-(motor_width/2-motor_screw_hole_position),-(motor_width/2-
            motor_screw_hole_position),0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([0,0,prolonging/2])
        triangular_prism_f(10,motor_width*4,motor_height/4);
    }
```

```
    translate([0,0,-prolonging/2])
    translate([sqrt(2)*motor_width/2-(screw_r+3)+3,0,-motor_height/2])
    difference(){
        cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
        translate([0,0,-fillet])
        cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    }
    translate([0,0,-prolonging/2])
    translate([-sqrt(2)*motor_width/2+(screw_r+3)-3,0,-motor_height/2])
    difference(){
        cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
        translate([0,0,-fillet])
        cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    }
    rotate([0,90,0])
    translate([0,0,sqrt(2)*motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
    rotate([0,-90,0])
    translate([0,0,sqrt(2)*motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
color([0.5,0.5,0.5])
translate([0,0,-prolonging/2])
union(){
    difference(){
    translate([0,0,-thickness-motor_height/2])
    cylinder_f(2,thickness,sqrt(2)*motor_width/2+thickness,sqrt(2)*motor_width/2+thickness);
    translate([sqrt(2)*motor_width/2-(screw_r+3)+3,0,-(motor_height+thickness+fillet*2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([sqrt(2)*motor_width/2-(screw_r+3)+3,0,prolonging/2-2-(motor_height+2*thickness
        +prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    translate([-sqrt(2)*motor_width/2+(screw_r+3)-3,0,-(motor_height+thickness+fillet*2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([-sqrt(2)*motor_width/2+(screw_r+3)-3,0,prolonging/2-2-(motor_height+2*
        thickness+prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    }
    rotate([180,0,0])
    translate([0,0,motor_height/2+thickness])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
```

```
}
}

if (display==1){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
color([0.5,0.5,0.5])
union(){
    translate([0,0,-prolonging/2])
    difference(){
        translate([0,0,-motor_height/2])  cylinder_f(5,motor_height+thickness+prolonging,sqrt
            (2)*motor_width/2+thickness,sqrt(2)*motor_width/2+thickness);
        translate([0,0,-(motor_height+thickness*2)/2-thickness])
        cylinder_f(5,motor_height+thickness*2,sqrt(2)*motor_width/2,sqrt(2)*motor_width/2);
        translate([0,0,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,motor_axis_offset_r+2,
            motor_axis_offset_r+2);
        rotate([0,0,-45])
        translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
        cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
        rotate([0,0,45])
        translate([-4*screw_r,-sqrt(2)*motor_width/2,motor_height/2+thickness])
        cube_f(4,8*screw_r,sqrt(2)*motor_width,2*prolonging);
        translate([0,0,(motor_height+prolonging+2*thickness)/2-connection_r/2])
        translate([0,0,prolonging/2])
        translate([-2*screw_r,-sqrt(2)*motor_width,-2*screw_r])
        cube_f(4,4*screw_r,sqrt(2)*motor_width*2,100*screw_r+thickness);
        translate([motor_width/2-motor_screw_hole_position,motor_width/2-
            motor_screw_hole_position,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([motor_width/2-motor_screw_hole_position,-(motor_width/2-
            motor_screw_hole_position),0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([-(motor_width/2-motor_screw_hole_position),motor_width/2-
            motor_screw_hole_position,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([-(motor_width/2-motor_screw_hole_position),-(motor_width/2-
            motor_screw_hole_position),0])
        cylinder(motor_height+thickness+ connection_r+ thickness,screw_r,screw_r);
        translate([0,0,prolonging/2])
        triangular_prism_f(10,motor_width*4,motor_height/4);
    }
    translate([0,0,-prolonging/2])
    translate([sqrt(2)*motor_width/2-(screw_r+3)+3,0,-motor_height/2])
    difference(){
        cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
        translate([0,0,-fillet])
        cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    }
    translate([0,0,-prolonging/2])
    translate([-sqrt(2)*motor_width/2+(screw_r+3)-3,0,-motor_height/2])
    difference(){
        cylinder_f(fillet,motor_height,screw_r+3,screw_r+3);
        translate([0,0,-fillet])
        cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    }
    rotate([0,90,0])
    translate([0,0,sqrt(2)*motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
    rotate([0,-90,0])
```

```
    translate([0,0,sqrt(2)*motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
}
if (display==2){
color([0.5,0.5,0.5])
translate([0,0,-prolonging/2])
union(){
    difference(){
    translate([0,0,-thickness-motor_height/2])
    cylinder_f(2,thickness,sqrt(2)*motor_width/2+thickness,sqrt(2)*motor_width/2+thickness);
    translate([sqrt(2)*motor_width/2-(screw_r+3)+3,0,-(motor_height+thickness+fillet*2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([sqrt(2)*motor_width/2-(screw_r+3)+3,0,prolonging/2-2-(motor_height+2*thickness
        +prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    translate([-sqrt(2)*motor_width/2+(screw_r+3)-3,0,-(motor_height+thickness+fillet*2)])
    cylinder(motor_height+thickness+fillet*2,screw_r,screw_r);
    translate([-sqrt(2)*motor_width/2+(screw_r+3)-3,0,prolonging/2-2-(motor_height+2*
        thickness+prolonging)/2])
    cylinder(4,screw_r+4,screw_r+4);
    }
    rotate([180,0,0])
    translate([0,0,motor_height/2+thickness])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }  }
    }
}

module motor_stand_f(display=true,cube_shape=false,motor_axis_offset_r=35,fillet=2,
    connection_h=20,connection_r=25,screw_r=3,motor_width=80,motor_height=123,
    motor_screw_hole_position=8.2,motor_axis_offset_h=3,motor_axis_length=32,thickness=8){
    if (cube_shape==false){ motor_stand_cylinder_f(display=display,motor_axis_offset_r=
        motor_axis_offset_r,fillet=fillet,connection_h=connection_h,connection_r=connection_r
        ,screw_r=screw_r,motor_width=motor_width,motor_height=motor_height,
        motor_screw_hole_position=motor_screw_hole_position,motor_axis_offset_h=
        motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=thickness);
        }
    if (cube_shape==true){ motor_stand_cube_f(display=display,motor_axis_offset_r=
        motor_axis_offset_r,fillet=fillet,connection_h=connection_h,connection_r=connection_r
        ,screw_r=screw_r,motor_width=motor_width,motor_height=motor_height,
        motor_screw_hole_position=motor_screw_hole_position,motor_axis_offset_h=
        motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=thickness);
        }
    }

//File name: motor_coupling.scad
//Purpose: generate a motor coupling.
//Parameter:
//1. fillet: the fillet radius.
//2. motor_height: the height of the motor.
```

```
//3. motor_axis_length: the length of the motor shaft.
//4. motor_axis_r: the radius of the motor shaft.
//5. motor_axis_offset_r: the radius of the motor mount hole.
//6. motor_axis_offset_h: the height of the motor mount hole.
//7. connection_h: the length of the connector.
//8. connection_r: the circumradius of the connector.
//9. thickness: wall thickness.
//10. screw_r: the radius of the screw hole.
include<cylinder_f.scad>
include<hexagon_prism_f.scad>
module motor_coupling(fillet = 2,motor_height = 123,motor_axis_length= 32,motor_axis_r= 9.5,
    motor_axis_offset_r = 35,motor_axis_offset_h = 3,connection_h=20,connection_r=27,
    thickness = 5,screw_r = 3){
translate([0,0,motor_height/2+motor_axis_offset_h])
color([0.8,0.8,0.8])
union(){
    translate([0,0,motor_axis_length])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
    translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
        )])
    cube(2*(connection_r+thickness));
    translate([0,0,thickness])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([0,0,thickness+connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r+thickness)])
    cylinder(4*(connection_r+thickness),screw_r,screw_r);
    }
    difference(){
        cylinder_f(fillet,motor_axis_length,motor_axis_offset_r,motor_axis_offset_r);
        translate([0,0,-fillet])
        cylinder(motor_axis_length+2*fillet,motor_axis_r,motor_axis_r);
    } } }
```

b) : Link.

```
//File name: link_f.scad
//Purpose: generate a Link.
//Parameter:
//1. cube_shape: joint shape settings. set to true for a cube shape, and set to false for a
    cylinder shape.
//2. motor_axis_offset_r: the radius of the motor mount hole.
//3. fillet: the fillet radius.
//4. connection_h: the length of the connector.
//5. connection_r: the circumradius of the connector.
//6. screw_r: the radius of the screw hole.
//7. motor_width: the width of the motor.
//8. motor_height: the height of the motor.
//9. motor_screw_hole_position: the shortest distance from the center of the screw hole to
    the edge.
//10. motor_axis_offset_h: the height of the motor mount hole.
//11. motor_axis_length: the length of the motor shaft.
//12. thickness: wall thickness.
include<hexagon_prism_f.scad>
include<cylinder_f.scad>
include<cube_f.scad>
include<triangular_prism_f.scad>
include<motor_coupling.scad>
module link_cube_f(motor_axis_offset_r=35,fillet=2,connection_h=20,connection_r=25,screw_r=3,
    motor_width=80,motor_height=123,motor_screw_hole_position=8.2,motor_axis_offset_h=3,
    motor_axis_length=32,thickness=8){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h - 2;
color([0.5,0.5,0.5])
union(){
    translate([0,0,-prolonging/2])
    difference(){
```

```
motor_coupling(fillet = fillet,motor_height = motor_height,motor_axis_length=
    motor_axis_length ,motor_axis_r= 9.5,motor_axis_offset_r = motor_axis_offset_r,
    motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
    connection_r,thickness = thickness,screw_r = screw_r);
cylinder_f(5,motor_height/2+motor_axis_offset_h +motor_axis_length,sqrt(2)*motor_width/2,
    sqrt(2)*motor_width/2);}
translate([0,0,-prolonging/2])
difference(){
    translate([0,0,-motor_height/2-thickness])
    translate([(-motor_width-2*thickness-(screw_r+3)*4)/2,-(motor_width+2*thickness+(
        screw_r+3)*4)/2,0])
    cube_f(5,(screw_r+3)*4+motor_width+2*thickness,(screw_r+3)*4+motor_width+2*thickness,
        motor_height+2*thickness+prolonging);
    translate([0,0,-motor_height/2])
    translate([-(motor_width+(screw_r+3)*4)/2,-(motor_width+(screw_r+3)*4)/2,0])
    cube_f(5,(screw_r+3)*4+motor_width,(screw_r+3)*4+motor_width,(motor_height+2*
        thickness+prolonging)-thickness-(connection_h+thickness-2));
    translate([0,0,0])
    cylinder(motor_height+thickness+ connection_r+ thickness,connection_r,connection_r);
    translate([0,0,(motor_height+prolonging+2*thickness)/2-connection_r/2])
    translate([0,0,prolonging/2])
    translate([-2*screw_r,-sqrt(2)*motor_width,-2*screw_r])
    cube_f(4,4*screw_r,sqrt(2)*motor_width*2,100*screw_r+thickness);
    translate([motor_width/2-thickness,motor_width/2-thickness,-(2*motor_height+
        prolonging+2*thickness)/2])
    rotate([0,0,45])
    translate([-7.5,-7.5,prolonging/2])
    cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
    translate([motor_width/2-thickness,-(motor_width/2-thickness),-(2*motor_height+
        prolonging+2*thickness)/2])
    rotate([0,0,45])
    translate([-7.5,-7.5,prolonging/2])
    cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
    translate([-(motor_width/2-thickness),motor_width/2-thickness,-(2*motor_height+
        prolonging+2*thickness)/2])
    rotate([0,0,45])
    translate([-7.5,-7.5,prolonging/2])
    cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
    translate([-(motor_width/2-thickness),-(motor_width/2-thickness),-(2*motor_height+
        prolonging+2*thickness)/2])
    rotate([0,0,45])
    translate([-7.5,-7.5,prolonging/2])
    cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
    translate([-((motor_height/2)/4)/2,-(motor_width*4)/2,prolonging/2-(motor_height/2)
        /2])
    cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);
    translate([-((motor_height/2)/4)/2+2*(motor_height/2)/4,-(motor_width*4)/2,prolonging
        /2-(motor_height/2)/2])
    cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);
    translate([-((motor_height/2)/4)/2-2*(motor_height/2)/4,-(motor_width*4)/2,prolonging
        /2-(motor_height/2)/2])
    cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);
    translate([0,0,prolonging/2+(motor_height+2*thickness+prolonging)/2-2*connection_h])
    union(){
    rotate([0,90,90])
    translate([-((motor_height/2)/4)/2,-(motor_width*4)/2,-(motor_height/2)/2])
    cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);}
    translate([0,0,prolonging/2-(motor_height+2*thickness+prolonging)/2+2*connection_h])
    union(){
    rotate([0,90,90])
    translate([-((motor_height/2)/4)/2,-(motor_width*4)/2,-(motor_height/2)/2])
    cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);}
}
rotate([0,90,0])
translate([0,0,(screw_r+3)*2+motor_width/2+thickness])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
```

```
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
    rotate([0,-90,0])
    translate([0,0,(screw_r+3)*2+motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
color([0.5,0.5,0.5])
translate([0,0,-prolonging/2])
union(){
    rotate([180,0,0])
    translate([0,0,motor_height/2+thickness])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
    }
}
module link_cylinder_f(motor_axis_offset_r=35,fillet=2,connection_h=20,connection_r=25,
    screw_r=3,motor_width=80,motor_height=123,motor_screw_hole_position=8.2,
    motor_axis_offset_h=3,motor_axis_length=32,thickness=8){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h - 2;
color([0.5,0.5,0.5])
union(){
    translate([0,0,-prolonging/2])
    difference(){
    motor_coupling(fillet = fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length ,motor_axis_r= 9.5,motor_axis_offset_r = motor_axis_offset_r,
        motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);
    cylinder_f(5,motor_height/2+motor_axis_offset_h +motor_axis_length,sqrt(2)*motor_width/2,
        sqrt(2)*motor_width/2);}
    translate([0,0,-prolonging/2])
    difference(){
        translate([0,0,-motor_height/2-thickness])
        cylinder_f(5,motor_height+2*thickness+prolonging,sqrt(2)*motor_width/2+thickness,sqrt
            (2)*motor_width/2+thickness);
        translate([0,0,prolonging/2-(motor_height+2*thickness+prolonging)/2+thickness])
        cylinder_f(5,(motor_height+2*thickness+prolonging)-thickness-(connection_h+thickness
            -2),sqrt(2)*motor_width/2,sqrt(2)*motor_width/2);
        translate([0,0,0])
        cylinder(motor_height+thickness+ connection_r+ thickness,connection_r,connection_r);
        translate([0,0,(motor_height+prolonging+2*thickness)/2-connection_r/2])
        translate([0,0,prolonging/2])
        translate([-2*screw_r,-sqrt(2)*motor_width,-2*screw_r])
        cube_f(4,4*screw_r,sqrt(2)*motor_width*2,100*screw_r+thickness);
        translate([motor_width/2-thickness,motor_width/2-thickness,-(2*motor_height+
            prolonging+2*thickness)/2])
        rotate([0,0,45])
```

```
        translate([-7.5,-7.5,prolonging/2])
        cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
        translate([motor_width/2-thickness,-(motor_width/2-thickness),-(2*motor_height+
            prolonging+2*thickness)/2])
        rotate([0,0,45])
        translate([-7.5,-7.5,prolonging/2])
        cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
        translate([-(motor_width/2-thickness),motor_width/2-thickness,-(2*motor_height+
            prolonging+2*thickness)/2])
        rotate([0,0,45])
        translate([-7.5,-7.5,prolonging/2])
        cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
        translate([-(motor_width/2-thickness),-(motor_width/2-thickness),-(2*motor_height+
            prolonging+2*thickness)/2])
        rotate([0,0,45])
        translate([-7.5,-7.5,prolonging/2])
        cube_f(5,15,15,2*motor_height+prolonging+2*thickness);
        translate([-((motor_height/2)/4)/2,-(motor_width*4)/2,prolonging/2-(motor_height/2)
            /2])
        cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);
        translate([-((motor_height/2)/4)/2+2*(motor_height/2)/4,-(motor_width*4)/2,prolonging
            /2-(motor_height/2)/2])
        cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);
        translate([-((motor_height/2)/4)/2-2*(motor_height/2)/4,-(motor_width*4)/2,prolonging
            /2-(motor_height/2)/2])
        cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);
        translate([0,0,prolonging/2+(motor_height+2*thickness+prolonging)/2-2*connection_h])
        union(){
        rotate([0,90,90])
        translate([-((motor_height/2)/4)/2,-(motor_width*4)/2,-(motor_height/2)/2])
        cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);}
        translate([0,0,prolonging/2-(motor_height+2*thickness+prolonging)/2+2*connection_h])
        union(){
        rotate([0,90,90])
        translate([-((motor_height/2)/4)/2,-(motor_width*4)/2,-(motor_height/2)/2])
        cube_f(5,(motor_height/2)/4,motor_width*4,motor_height/2);}
    }
    rotate([0,90,0])
    translate([0,0,sqrt(2)*motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
    rotate([0,-90,0])
    translate([0,0,sqrt(2)*motor_width/2+thickness])
    difference(){
    translate([0,0,-fillet-thickness])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)-thickness])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
color([0.5,0.5,0.5])
translate([0,0,-prolonging/2])
union(){
    rotate([180,0,0])
    translate([0,0,motor_height/2+thickness])
    difference(){
```

```
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([-(connection_r),-(connection_r),-2*(connection_r)])
    cube(2*(connection_r));
    translate([0,0,connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r)])
    cylinder(4*(connection_r),screw_r,screw_r);
    }
}
}
module link_f(cube_shape=false,motor_axis_offset_r=35,fillet=2,connection_h=20,connection_r
    =25,screw_r=3,motor_width=80,motor_height=123,motor_screw_hole_position=8.2,
    motor_axis_offset_h=3,motor_axis_length=32,thickness=8){
    if (cube_shape==false){link_cylinder_f(motor_axis_offset_r=motor_axis_offset_r,fillet=
        fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,
        motor_width=motor_width,motor_height=motor_height,motor_screw_hole_position=
        motor_screw_hole_position,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=
        motor_axis_length,thickness=thickness);
        }
    if (cube_shape==true){link_cube_f(motor_axis_offset_r=motor_axis_offset_r,fillet=fillet,
        connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
        motor_width,motor_height=motor_height,motor_screw_hole_position=
        motor_screw_hole_position,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=
        motor_axis_length,thickness=thickness);
        }
    }
```

c) : Chassis.

```
//File name: chassis_f.scad
//Purpose: generate a Chassis.
//Parameter:
//1. display: display settings. when set to true, display the entire assembled module; when
    set to a specific number, the corresponding number of parts is displayed.
//2. fillet: the fillet radius.
//3. wheel_track: the width of the chassis.
//4. chassis_length: the length of the chassis.
//5. chassis_thickness: the height of the chassis.
//6. connection_r: the circumradius of the connector.
//7. connection_h: the length of the connector.
//8. thickness: wall thickness.
//9. screw_r: the radius of the screw hole.
//10. container_width: the width of the Joint.
//11. container_height: the height of the Joint.
include<trapezoid_prism_f.scad>
include<cube_f.scad>
include<cylinder_f.scad>
include<hexagon_prism_f.scad>
module chassis_f(display=true,fillet=2,wheel_track=250,chassis_length=200,chassis_thickness
    =100,connection_r=25,connection_h=20,thickness=8,screw_r=3,container_width=100,
    container_height=100){
if (display==true){
color([0.7,0.7,0.7])
union(){
    rotate([0,90,0])
    translate([0,0,-thickness-connection_h])
    translate([0,0,wheel_track/2+2])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
    translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
        )])
    cube(2*(connection_r+thickness));
    translate([0,0,thickness])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([0,0,thickness+connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r+thickness)])
```

```
cylinder(4*(connection_r+thickness),screw_r,screw_r);
}
rotate([0,-90,0])
translate([0,0,-thickness-connection_h])
translate([0,0,wheel_track/2+2])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
    )])
cube(2*(connection_r+thickness));
translate([0,0,thickness])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([0,0,thickness+connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r+thickness)])
cylinder(4*(connection_r+thickness),screw_r,screw_r);
}
difference(){
    translate([-(wheel_track)/2,-(chassis_length+20)/2,-(chassis_thickness)/2])
    difference(){
        cube_f(5,wheel_track,chassis_length+20,chassis_thickness-(thickness+connection_h
            -2));
        translate([(thickness+connection_h-2),-thickness,(thickness+connection_h-2)])
        cube_f(5,wheel_track-2*(thickness+connection_h-2),chassis_length+2*thickness+20,
            chassis_thickness-2*thickness);}
    rotate([0,90,0])
    translate([0,0,-wheel_track])
    cylinder(wheel_track*2,connection_r,connection_r);
    rotate([0,90,0])
    translate([0,0,wheel_track/2+2-connection_h/2])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
    rotate([0,-90,0])
    translate([0,0,wheel_track/2+2-connection_h/2])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
    }
translate([wheel_track/2-(thickness+connection_h-2)-screw_r,0,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([-(wheel_track/2-(thickness+connection_h-2)-screw_r),0,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([0,-chassis_length+(thickness+connection_h-2)+screw_r,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([0,chassis_length-(thickness+connection_h-2)-screw_r,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
difference(){
```

```
    trapezoid_prism_f(5,(wheel_track)/2,wheel_track,(chassis_length)/2,chassis_thickness
        -(thickness+connection_h-2));
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness
        -2*(thickness+connection_h-2));
    translate([0,-(chassis_length-(thickness+connection_h-2)),0])
    translate([-((wheel_track)+container_height*2),(chassis_length)/2,(thickness+
        connection_h-2)/2-(chassis_thickness/2-(thickness+connection_h-2))])
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness);
}
rotate([0,0,180])
translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
difference(){
   trapezoid_prism_f(5,(wheel_track)/2,(wheel_track),(chassis_length)/2,chassis_thickness
       -(thickness+connection_h-2));
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness
        -2*(thickness+connection_h-2));
    translate([0,-(chassis_length-(thickness+connection_h-2)),0])
    translate([-((wheel_track)+container_height*2),(chassis_length)/2,(thickness+
        connection_h-2)/2-(chassis_thickness/2-(thickness+connection_h-2))])
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness);
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([-90,0,0])
    translate([-(wheel_track)/8,0,-((chassis_length)*4)])
    cylinder((chassis_length)*8,connection_r,connection_r);
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([-90,0,0])
    translate([(wheel_track)/8,0,-((chassis_length)*4)])
    cylinder((chassis_length)*8,connection_r,connection_r);
    translate([-(wheel_track)/8,-chassis_length-2+connection_h/2,0])
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([90,0,0])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
    translate([(wheel_track)/8,-chassis_length-2+connection_h/2,0])
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([90,0,0])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
}
rotate([90,0,0])
translate([(wheel_track)/8,0,chassis_length])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([-(connection_r),-(connection_r),-2*(connection_r)])
cube(2*(connection_r));
translate([0,0,connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r)])
cylinder(4*(connection_r),screw_r,screw_r);
}
rotate([90,0,0])
translate([-(wheel_track)/8,0,chassis_length])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);

translate([-(connection_r),-(connection_r),-2*(connection_r)])
cube(2*(connection_r));
translate([0,0,connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r)])
cylinder(4*(connection_r),screw_r,screw_r);
}
rotate([-90,0,0])
translate([(wheel_track)/8,0,chassis_length-(thickness+connection_h-2)])
difference(){
    translate([0,0,-fillet])
```

```
        hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
        translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+
            thickness)])
        cube(2*(connection_r+thickness));
        translate([0,0,thickness])
        hexagon_prism_f(fillet,connection_h+fillet,connection_r);
        translate([0,0,thickness+connection_h/2])
        rotate([90,0,0])
        translate([0,0,-2*(connection_r+thickness)])
        cylinder(4*(connection_r+thickness),screw_r,screw_r);
    }
    rotate([-90,0,0])
    translate([-(wheel_track)/8,0,chassis_length-(thickness+connection_h-2)])
    difference(){
        translate([0,0,-fillet])
        hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
        translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+
            thickness)])
        cube(2*(connection_r+thickness));
        translate([0,0,thickness])
        hexagon_prism_f(fillet,connection_h+fillet,connection_r);
        translate([0,0,thickness+connection_h/2])
        rotate([90,0,0])
        translate([0,0,-2*(connection_r+thickness)])
        cylinder(4*(connection_r+thickness),screw_r,screw_r);
    }
}
color([0.7,0.7,0.7])
union(){
    difference(){
        union(){
        translate([-(wheel_track)/2,-(chassis_length+20)/2,chassis_thickness/2-(thickness+
            connection_h-2)])
        cube_f(5,wheel_track,chassis_length+20,(thickness+connection_h-2));
        translate([0,-chassis_length/2,chassis_thickness/2-(thickness+connection_h-2)/2])
        trapezoid_prism_f(5,(wheel_track)/2,(wheel_track),(chassis_length)/2,(thickness+
            connection_h-2));
        rotate([0,0,180])
        translate([0,-chassis_length/2,chassis_thickness/2-(thickness+connection_h-2)/2])
        trapezoid_prism_f(5,(wheel_track)/2,(wheel_track),(chassis_length)/2,(thickness+
            connection_h-2));
        }
        translate([wheel_track/2-(thickness+connection_h-2)-screw_r,0,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([wheel_track/2-(thickness+connection_h-2)-screw_r,0,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([-(wheel_track/2-(thickness+connection_h-2)-screw_r),0,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([-(wheel_track/2-(thickness+connection_h-2)-screw_r),0,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([0,-chassis_length+(thickness+connection_h-2)+screw_r,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([0,-chassis_length+(thickness+connection_h-2)+screw_r,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([0,chassis_length-(thickness+connection_h-2)-screw_r,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([0,chassis_length-(thickness+connection_h-2)-screw_r,0])
```

```
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([0,-((container_width+2*thickness)+thickness),0])
        translate([-(wheel_track/2)/2,-(chassis_length-(thickness+connection_h-2)-(
            container_width+2*thickness)-4*thickness),0])
        cube_f(5,wheel_track/2,chassis_length-(thickness+connection_h-2)-(container_width+2*
            thickness)-4*thickness,chassis_thickness/2+2*thickness);
        rotate([0,0,180])
        translate([0,-((container_width+2*thickness)+thickness),0])
        translate([-(wheel_track/2)/2,-(chassis_length-(thickness+connection_h-2)-(
            container_width+2*thickness)-4*thickness),0])
        cube_f(5,wheel_track/2,chassis_length-(thickness+connection_h-2)-(container_width+2*
            thickness)-4*thickness,chassis_thickness/2+2*thickness);
        rotate([0,0,180])
        translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
        translate([-(wheel_track)/8,-chassis_length-2+connection_h/2,0])
        translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
        rotate([90,0,0])
        translate([-2*screw_r,-chassis_length,-2*screw_r])
        cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
        rotate([0,0,180])
        translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
        translate([(wheel_track)/8,-chassis_length-2+connection_h/2,0])
        translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
        rotate([90,0,0])
        translate([-2*screw_r,-chassis_length,-2*screw_r])
        cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
        }
    difference(){
        translate([0,0,chassis_thickness/2-5])
        cylinder_f(5,thickness+connection_h-2,container_width+2*thickness,container_width+2*
            thickness);
        translate([0,0,chassis_thickness/2-5])
        translate([0,0,-thickness])
        cylinder_f(5,thickness+connection_h-2+2*thickness,connection_r,connection_r);
        translate([0,0,chassis_thickness/2+thickness+connection_h-2-5-connection_h/2])
        translate([-2*screw_r,-chassis_length,-2*screw_r])
        cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
        }
    translate([0,0,chassis_thickness/2-5])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
    translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
        )])
    cube(2*(connection_r+thickness));
    translate([0,0,thickness])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([0,0,thickness+connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r+thickness)])
    cylinder(4*(connection_r+thickness),screw_r,screw_r);
    }
}
}
if (display==1){
color([0.7,0.7,0.7])
union(){
    rotate([0,90,0])
    translate([0,0,-thickness-connection_h])
    translate([0,0,wheel_track/2+2])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
    translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
        )])
    cube(2*(connection_r+thickness));
    translate([0,0,thickness])
```

```
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([0,0,thickness+connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r+thickness)])
cylinder(4*(connection_r+thickness),screw_r,screw_r);
}
rotate([0,-90,0])
translate([0,0,-thickness-connection_h])
translate([0,0,wheel_track/2+2])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
    )])
cube(2*(connection_r+thickness));
translate([0,0,thickness])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([0,0,thickness+connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r+thickness)])
cylinder(4*(connection_r+thickness),screw_r,screw_r);
}
difference(){
    translate([-(wheel_track)/2,-(chassis_length+20)/2,-(chassis_thickness)/2])
    difference(){
        cube_f(5,wheel_track,chassis_length+20,chassis_thickness-(thickness+connection_h
            -2));
        translate([(thickness+connection_h-2),-thickness,(thickness+connection_h-2)])
        cube_f(5,wheel_track-2*(thickness+connection_h-2),chassis_length+2*thickness+20,
            chassis_thickness-2*thickness);}
    rotate([0,90,0])
    translate([0,0,-wheel_track])
    cylinder(wheel_track*2,connection_r,connection_r);
    rotate([0,90,0])
    translate([0,0,wheel_track/2+2-connection_h/2])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
    rotate([0,-90,0])
    translate([0,0,wheel_track/2+2-connection_h/2])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
    }
translate([wheel_track/2-(thickness+connection_h-2)-screw_r,0,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([-(wheel_track/2-(thickness+connection_h-2)-screw_r),0,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([0,-chassis_length+(thickness+connection_h-2)+screw_r,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([0,chassis_length-(thickness+connection_h-2)-screw_r,-chassis_thickness/2])
difference(){
    cylinder_f(fillet,chassis_thickness-(thickness+connection_h-2),screw_r+6,screw_r+6);
    translate([0,0,-fillet])
```

```
    cylinder(chassis_thickness-(thickness+connection_h-2)+thickness+fillet*2,screw_r,
        screw_r);
}
translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
difference(){
    trapezoid_prism_f(5,(wheel_track)/2,wheel_track,(chassis_length)/2,chassis_thickness
        -(thickness+connection_h-2));
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness
        -2*(thickness+connection_h-2));
    translate([0,-(chassis_length-(thickness+connection_h-2)),0])
    translate([-((wheel_track)+container_height*2),(chassis_length)/2,(thickness+
        connection_h-2)/2-(chassis_thickness/2-(thickness+connection_h-2))])
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness);
}
rotate([0,0,180])
translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
difference(){
   trapezoid_prism_f(5,(wheel_track)/2,(wheel_track),(chassis_length)/2,chassis_thickness
       -(thickness+connection_h-2));
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness
        -2*(thickness+connection_h-2));
    translate([0,-(chassis_length-(thickness+connection_h-2)),0])
    translate([-((wheel_track)+container_height*2),(chassis_length)/2,(thickness+
        connection_h-2)/2-(chassis_thickness/2-(thickness+connection_h-2))])
    cube_f(5,2*((wheel_track)+container_height*2),(chassis_length)*2,chassis_thickness);
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([-90,0,0])
    translate([-(wheel_track)/8,0,-((chassis_length)*4)])
    cylinder((chassis_length)*8,connection_r,connection_r);
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([-90,0,0])
    translate([(wheel_track)/8,0,-((chassis_length)*4)])
    cylinder((chassis_length)*8,connection_r,connection_r);
    translate([-(wheel_track)/8,-chassis_length-2+connection_h/2,0])
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([90,0,0])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
    translate([(wheel_track)/8,-chassis_length-2+connection_h/2,0])
    translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
    rotate([90,0,0])
    translate([-2*screw_r,-chassis_length,-2*screw_r])
    cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
}
rotate([90,0,0])
translate([(wheel_track)/8,0,chassis_length])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([-(connection_r),-(connection_r),-2*(connection_r)])
cube(2*(connection_r));
translate([0,0,connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r)])
cylinder(4*(connection_r),screw_r,screw_r);
}
rotate([90,0,0])
translate([-(wheel_track)/8,0,chassis_length])
difference(){
translate([0,0,-fillet])
hexagon_prism_f(fillet,connection_h+fillet,connection_r);
translate([-(connection_r),-(connection_r),-2*(connection_r)])
cube(2*(connection_r));
translate([0,0,connection_h/2])
rotate([90,0,0])
translate([0,0,-2*(connection_r)])
cylinder(4*(connection_r),screw_r,screw_r);
}
```

```
    rotate([-90,0,0])
    translate([(wheel_track)/8,0,chassis_length-(thickness+connection_h-2)])
    difference(){
        translate([0,0,-fillet])
        hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
        translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+
            thickness)])
        cube(2*(connection_r+thickness));
        translate([0,0,thickness])
        hexagon_prism_f(fillet,connection_h+fillet,connection_r);
        translate([0,0,thickness+connection_h/2])
        rotate([90,0,0])
        translate([0,0,-2*(connection_r+thickness)])
        cylinder(4*(connection_r+thickness),screw_r,screw_r);
    }
    rotate([-90,0,0])
    translate([-(wheel_track)/8,0,chassis_length-(thickness+connection_h-2)])
    difference(){
        translate([0,0,-fillet])
        hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
        translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+
            thickness)])
        cube(2*(connection_r+thickness));
        translate([0,0,thickness])
        hexagon_prism_f(fillet,connection_h+fillet,connection_r);
        translate([0,0,thickness+connection_h/2])
        rotate([90,0,0])
        translate([0,0,-2*(connection_r+thickness)])
        cylinder(4*(connection_r+thickness),screw_r,screw_r);
    } } }

if(display==2){
color([0.7,0.7,0.7])
union(){
    difference(){
        union(){
        translate([-(wheel_track)/2,-(chassis_length+20)/2,chassis_thickness/2-(thickness+
            connection_h-2)])
        cube_f(5,wheel_track,chassis_length+20,(thickness+connection_h-2));
        translate([0,-chassis_length/2,chassis_thickness/2-(thickness+connection_h-2)/2])
        trapezoid_prism_f(5,(wheel_track)/2,(wheel_track),(chassis_length)/2,(thickness+
            connection_h-2));
        rotate([0,0,180])
        translate([0,-chassis_length/2,chassis_thickness/2-(thickness+connection_h-2)/2])
        trapezoid_prism_f(5,(wheel_track)/2,(wheel_track),(chassis_length)/2,(thickness+
            connection_h-2));
        }
        translate([wheel_track/2-(thickness+connection_h-2)-screw_r,0,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([wheel_track/2-(thickness+connection_h-2)-screw_r,0,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([-(wheel_track/2-(thickness+connection_h-2)-screw_r),0,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([-(wheel_track/2-(thickness+connection_h-2)-screw_r),0,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([0,-chassis_length+(thickness+connection_h-2)+screw_r,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([0,-chassis_length+(thickness+connection_h-2)+screw_r,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
```

```
        translate([0,chassis_length-(thickness+connection_h-2)-screw_r,0])
        translate([0,0,(chassis_thickness/2-(thickness+connection_h-2)/2)])
        translate([0,0,-(thickness+connection_h-2)])
        cylinder(2*(thickness+connection_h-2),screw_r,screw_r);
        translate([0,chassis_length-(thickness+connection_h-2)-screw_r,0])
        translate([0,0,chassis_thickness/2-(thickness+connection_h-2)+thickness])
        cylinder(2*(thickness+connection_h-2),screw_r+4,screw_r+4);
        translate([0,-((container_width+2*thickness)+thickness),0])
        translate([-(wheel_track/2)/2,-(chassis_length-(thickness+connection_h-2)-(
            container_width+2*thickness)-4*thickness),0])
        cube_f(5,wheel_track/2,chassis_length-(thickness+connection_h-2)-(container_width+2*
            thickness)-4*thickness,chassis_thickness/2+2*thickness);
        rotate([0,0,180])
        translate([0,-((container_width+2*thickness)+thickness),0])
        translate([-(wheel_track/2)/2,-(chassis_length-(thickness+connection_h-2)-(
            container_width+2*thickness)-4*thickness),0])
        cube_f(5,wheel_track/2,chassis_length-(thickness+connection_h-2)-(container_width+2*
            thickness)-4*thickness,chassis_thickness/2+2*thickness);
        rotate([0,0,180])
        translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
        translate([-(wheel_track)/8,-chassis_length-2+connection_h/2,0])
        translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
        rotate([90,0,0])
        translate([-2*screw_r,-chassis_length,-2*screw_r])
        cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
        rotate([0,0,180])
        translate([0,-(chassis_length)/2,-(thickness+connection_h-2)/2])
        translate([(wheel_track)/8,-chassis_length-2+connection_h/2,0])
        translate([0,(chassis_length)/2,(thickness+connection_h-2)/2])
        rotate([90,0,0])
        translate([-2*screw_r,-chassis_length,-2*screw_r])
        cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
        }
    difference(){
        translate([0,0,chassis_thickness/2-5])
        cylinder_f(5,thickness+connection_h-2,container_width+2*thickness,container_width+2*
            thickness);
        translate([0,0,chassis_thickness/2-5])
        translate([0,0,-thickness])
        cylinder_f(5,thickness+connection_h-2+2*thickness,connection_r,connection_r);
        translate([0,0,chassis_thickness/2+thickness+connection_h-2-5-connection_h/2])
        translate([-2*screw_r,-chassis_length,-2*screw_r])
        cube_f(4,4*screw_r,2*chassis_length,100*screw_r+thickness);
        }
    translate([0,0,chassis_thickness/2-5])
    difference(){
    translate([0,0,-fillet])
    hexagon_prism_f(fillet,connection_h+fillet+thickness,connection_r+thickness);
    translate([-(connection_r+thickness),-(connection_r+thickness),-2*(connection_r+thickness
        )])
    cube(2*(connection_r+thickness));
    translate([0,0,thickness])
    hexagon_prism_f(fillet,connection_h+fillet,connection_r);
    translate([0,0,thickness+connection_h/2])
    rotate([90,0,0])
    translate([0,0,-2*(connection_r+thickness)])
    cylinder(4*(connection_r+thickness),screw_r,screw_r);
    } } }
}
```

d) : Wheel.

```
//File name: wheel_f.scad
//Purpose: generate a Wheel.
//Parameter:
//1. fillet: the fillet radius.
//2. wheel_thickness: the thickness of the wheel.
//3. wheel_r: the radius of the wheel.
//4. connection_h: the length of the connector.
```

```
//5. connection_r: the circumradius of the connector.
//6. screw_r: the radius of the screw hole.
include<cylinder_f.scad>
include<hexagon_prism_f.scad>
include<cube_f.scad>
module stl_original(){
    import("tire_v3.stl");
    }
module wheel_f(fillet=2,wheel_thickness=80,wheel_r=100,connection_h=20,connection_r=25,
    screw_r=3){
    color([0.3,0.3,0.3])
    rotate([0,-90,0])
    translate([0,0,-wheel_thickness/2])
    union(){
    translate([0,0,wheel_thickness])
    difference(){
        translate([0,0,-fillet])
        hexagon_prism_f(fillet,connection_h+fillet,connection_r);
        translate([-(connection_r),-(connection_r),-2*(connection_r)])
        cube(2*(connection_r));
        translate([0,0,connection_h/2])
        rotate([90,0,0])
        translate([0,0,-2*(connection_r)])
        cylinder(4*(connection_r),screw_r,screw_r);
     }
    for (i=[0:1:7]){
        rotate([0,0,i*360/8])
        translate([0,-(20)/2,0])
        cube_f(5,wheel_r/1.5,20,wheel_thickness);
        }
    cylinder_f(5,wheel_thickness,wheel_r/3,wheel_r/3);
    scale([wheel_r/26,wheel_r/26,wheel_thickness/13.6])
    stl_original();
    }
}
```

C. *Supplementary Modules*

Before writing the files for supplementary modules such as Leg, Leg-wheel, and Mecanum-wheel, include the above-mentioned geometries. Use these geometries along with OpenSCAD's built-in functions to create the supplementary modules (Mecanum-wheel refers to roller.stl and wheel-hub.stl). The OpenSCAD code is as follows:

a) : Leg.

```
//File name: leg.scad
//Purpose: generate a Leg.
//Parameter:
//1. display: display settings. when set to true, display the entire assembled module; when
    set to a specific number, the corresponding number of parts is displayed.
//2. fillet: the fillet radius.
//3. cube_shape: joint shape settings. set to true for a cube shape, and set to false for a
    cylinder shape.
//4. thigh_length: the length of the thigh.
//5. shank_length: the length of the shank.
//6. motor_width: the width of the motor.
//7. motor_height: the height of the motor.
//8. motor_axis_r: the radius of the motor shaft.
//9. motor_axis_offset_r: the radius of the motor mount hole.
//10. motor_screw_hole_position: the shortest distance from the center of the screw hole to
    the edge.
//11. motor_axis_offset_h: the height of the motor mount hole.
//12. motor_axis_length: the length of the motor shaft.
//13. connection_h: the length of the connector.
//14. connection_r: the circumradius of the connector.
//15. screw_r: the radius of the screw hole.
//16. thickness: wall thickness.
include<cube_f.scad>
include<cylinder_f.scad>
include<motor_stand_f.scad>
```

```
include<motor_coupling.scad>
include<male_connector.scad>
include<female_connector.scad>
module leg(display=true,fillet=2,cube_shape=false,thigh_length,shank_length,motor_width,
    motor_height,motor_axis_r,motor_axis_offset_r,motor_screw_hole_position,
    motor_axis_offset_h,motor_axis_length,connection_h,connection_r,screw_r,thickness){
if(display==true){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
union(){
motor_stand_f(cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=fillet,
    connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
translate([0,0,-prolonging/2])
{
    motor_coupling(fillet =fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length,motor_axis_r=motor_axis_r,motor_axis_offset_r = motor_axis_offset_r
        ,motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);}
}
color([0.5,0.5,0.5])
difference(){
rotate([-135,0,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)/2])
cube_f((motor_height+prolonging+2*thickness)/8,(motor_height+prolonging+2*thickness)/2,
    thigh_length,(motor_height+prolonging+2*thickness)/2);
translate([(motor_height+prolonging+2*thickness)/4,0,0])
rotate([0,90,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
cylinder((motor_height+prolonging+2*thickness),(sqrt(2)*motor_width+2*thickness)/2,(sqrt(2)*
    motor_width+2*thickness)/2) ;
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    translate([(motor_height+prolonging+2*thickness)/4,0,0])
    rotate([0,90,0])
    translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
    cylinder(h=(motor_height+prolonging+2*thickness),r=connection_r/2);
    rotate([-135,0,0])
    translate([-(motor_height+prolonging+2*thickness)/4,0,-((motor_height+prolonging+2*
        thickness)/4)/2])
    cube([(motor_height+prolonging+2*thickness),thigh_length-4*connection_r,(motor_height+
        prolonging+2*thickness)/4]);
}
color([0.4,0.4,0.4])
translate([(motor_height+prolonging+2*thickness)/2,-(thigh_length-2*connection_r)/sqrt(2),-(
    thigh_length-2*connection_r)/sqrt(2)])
rotate([-45,0,0])
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,shank_length+5*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}
color([0.4,0.4,0.4])
difference(){
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,0])
    cylinder_f(5,(motor_height+prolonging+2*thickness)/2+2*connection_r,connection_r/2,
        connection_r/2);
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
```

```
    translate([0,0,-1*connection_r])    cylinder((motor_height+prolonging+2*thickness)/2+4*
        connection_r,screw_r,screw_r);
    }
color([0.8,0.8,0.8])
translate([(motor_height+prolonging+2*thickness)/2-(connection_r),-(4*connection_r)/sqrt(2)
    ,(4*connection_r)/sqrt(2)])
translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt(2)])
rotate([45,0,0])
union(){
difference(){
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r);
    }
difference(){
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r); }
translate([0,0,-connection_r/2])
cube_f(connection_r/4,connection_r,thigh_length-(2*connection_r),connection_r);}
color([1,1,1])
union(){
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([-45,0,0])
union(){
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,6*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}
rotate([0,90,0])
cylinder_f(2*connection_r/10,2*connection_r,connection_r+thickness,connection_r+thickness);
}
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([0,-90,0])
male_connector(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r);
}
color([0.4,0.4,0.4])
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))-(
    shank_length/sqrt(2))+2*connection_r])
sphere(2*connection_r);
}
if(display==1){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
motor_stand_f(display=1,cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=
    fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
color([0.5,0.5,0.5])
difference(){
rotate([-135,0,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)/2])
cube_f((motor_height+prolonging+2*thickness)/8,(motor_height+prolonging+2*thickness)/2,
    thigh_length,(motor_height+prolonging+2*thickness)/2);
translate([(motor_height+prolonging+2*thickness)/4,0,0])
rotate([0,90,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
cylinder((motor_height+prolonging+2*thickness),(sqrt(2)*motor_width+2*thickness)/2,(sqrt(2)*
    motor_width+2*thickness)/2) ;
```

```
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    translate([(motor_height+prolonging+2*thickness)/4,0,0])
    rotate([0,90,0])
    translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
    cylinder(h=(motor_height+prolonging+2*thickness),r=connection_r/2);
    rotate([-135,0,0])
    translate([-(motor_height+prolonging+2*thickness)/4,0,-((motor_height+prolonging+2*
        thickness)/4)/2])
    cube([(motor_height+prolonging+2*thickness),thigh_length-4*connection_r,(motor_height+
        prolonging+2*thickness)/4]);
    }
    }
if(display==2){
    prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
motor_stand_f(display=2,cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=
    fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
    }
if(display==3){
    prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
union(){
translate([0,0,-prolonging/2])
{
    ser_motor_f();}
}
    }
if(display==4){
    prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
union(){
translate([0,0,-prolonging/2])
{
    motor_coupling(fillet =fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length,motor_axis_r=motor_axis_r,motor_axis_offset_r = motor_axis_offset_r
        ,motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);}
}
    }
if(display==5){
color([1,1,1])
union(){
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([-45,0,0])
union(){
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,6*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}
rotate([0,90,0])
cylinder_f(2*connection_r/10,2*connection_r,connection_r+thickness,connection_r+thickness);
}
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([0,-90,0])
male_connector(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r);
}
    }
if(display==6){
color([0.8,0.8,0.8])
```

```
translate([(motor_height+prolonging+2*thickness)/2-(connection_r),-(4*connection_r)/sqrt(2)
    ,(4*connection_r)/sqrt(2)])
translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt(2)])
rotate([45,0,0])
union(){
difference(){
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r);
    }
difference(){
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r); }
translate([0,0,-connection_r/2])
cube_f(connection_r/4,connection_r,thigh_length-(2*connection_r),connection_r);}
    }
if(display==7){
color([0.4,0.4,0.4])
translate([(motor_height+prolonging+2*thickness)/2,-(thigh_length-2*connection_r)/sqrt(2),-(
    thigh_length-2*connection_r)/sqrt(2)])
rotate([-45,0,0])
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,shank_length+5*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}
color([0.4,0.4,0.4])
difference(){
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,0])
    cylinder_f(5,(motor_height+prolonging+2*thickness)/2+2*connection_r,connection_r/2,
        connection_r/2);
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,-1*connection_r])  cylinder((motor_height+prolonging+2*thickness)/2+4*
        connection_r,screw_r,screw_r);
    }
color([0.4,0.4,0.4])
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))-(
    shank_length/sqrt(2))+2*connection_r])
sphere(2*connection_r);
    }
}
```

b) : Leg-wheel.

```
//File name: leg_wheel.scad
//Purpose: generate a Leg-wheel.
//Parameter:
//1. display: display settings. when set to true, display the entire assembled module; when
    set to a specific number, the corresponding number of parts is displayed.
//2. fillet: the fillet radius.
//3. cube_shape: joint shape settings. set to true for a cube shape, and set to false for a
    cylinder shape.
//4. thigh_length: the length of the thigh.
//5. shank_length: the length of the shank.
//6. motor_width: the width of the motor.
//7. motor_height: the height of the motor.
```

```
//8. motor_axis_r: the radius of the motor shaft.
//9. motor_axis_offset_r: the radius of the motor mount hole.
//10. motor_screw_hole_position: the shortest distance from the center of the screw hole to
    the edge.
//11. motor_axis_offset_h: the height of the motor mount hole.
//12. motor_axis_length: the length of the motor shaft.
//13. connection_h: the length of the connector.
//14. connection_r: the circumradius of the connector.
//15. screw_r: the radius of the screw hole.
//16. thickness: wall thickness.
//17. wheel_thickness: the thickness of the wheel.
//18. wheel_r: the radius of the wheel.
include<cube_f.scad>
include<cylinder_f.scad>
include<motor_stand_f.scad>
include<motor_coupling.scad>
include<male_connector.scad>
include<female_connector.scad>
include<wheel_f.scad>
module leg_wheel(display=true,fillet=2,cube_shape=false,thigh_length,shank_length,motor_width
    ,motor_height,motor_axis_r,motor_axis_offset_r,motor_screw_hole_position,
    motor_axis_offset_h,motor_axis_length,connection_h,connection_r,screw_r,thickness,
    wheel_thickness,wheel_r){
if(display==true){
rotate([0,90,0])
union(){
motor_stand_f(cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=fillet,
    connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
translate([0,0,-prolonging/2])
{
    motor_coupling(fillet =fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length,motor_axis_r=motor_axis_r,motor_axis_offset_r = motor_axis_offset_r
        ,motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);}
}
color([0.5,0.5,0.5])
difference(){
rotate([-135,0,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)/2])
cube_f((motor_height+prolonging+2*thickness)/8,(motor_height+prolonging+2*thickness)/2,
    thigh_length,(motor_height+prolonging+2*thickness)/2);
translate([(motor_height+prolonging+2*thickness)/4,0,0])
rotate([0,90,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
cylinder((motor_height+prolonging+2*thickness),(sqrt(2)*motor_width+2*thickness)/2,(sqrt(2)*
    motor_width+2*thickness)/2);
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    translate([(motor_height+prolonging+2*thickness)/4,0,0])
    rotate([0,90,0])
    translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
    cylinder(h=(motor_height+prolonging+2*thickness),r=connection_r/2);
    rotate([-135,0,0])
    translate([-(motor_height+prolonging+2*thickness)/4,0,-((motor_height+prolonging+2*
        thickness)/4)/2])
    cube([(motor_height+prolonging+2*thickness),thigh_length-4*connection_r,(motor_height+
        prolonging+2*thickness)/4]);
}
color([0.4,0.4,0.4])
difference(){
union(){
translate([(motor_height+prolonging+2*thickness)/2,-(thigh_length-2*connection_r)/sqrt(2),-(
    thigh_length-2*connection_r)/sqrt(2)])
rotate([-45,0,0])
difference(){
```

```
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,shank_length+5*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}}
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,0])
translate([0,0,-(thigh_length/sqrt(2))-(shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
cylinder((motor_height+prolonging+2*thickness),(sqrt(2)*motor_width+2*thickness)/2,(sqrt(2)*
    motor_width+2*thickness)/2) ;
}
color([0.4,0.4,0.4])
difference(){
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,0])
    cylinder_f(5,(motor_height+prolonging+2*thickness)/2+2*connection_r,connection_r/2,
        connection_r/2);
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,-1*connection_r])   cylinder((motor_height+prolonging+2*thickness)/2+4*
        connection_r,screw_r,screw_r);
    }
color([0.8,0.8,0.8])
translate([(motor_height+prolonging+2*thickness)/2-(connection_r),-(4*connection_r)/sqrt(2)
    ,(4*connection_r)/sqrt(2)])
translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt(2)])
rotate([45,0,0])
union(){
difference(){
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);

translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r);
    }
difference(){
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r);
    }
translate([0,0,-connection_r/2])
cube_f(connection_r/4,connection_r,thigh_length-(2*connection_r),connection_r);}
color([1,1,1])
union(){
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([-45,0,0])
union(){
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,6*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}
rotate([0,90,0])
cylinder_f(2*connection_r/10,2*connection_r,connection_r+thickness,connection_r+thickness);
}
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
```

```
rotate([0,-90,0])
male_connector(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r);
}
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))-(
    shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
union(){
motor_stand_f(cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=fillet,
    connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
translate([0,0,-prolonging/2])
{
    motor_coupling(fillet =fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length,motor_axis_r=motor_axis_r,motor_axis_offset_r = motor_axis_offset_r
        ,motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);}
}
translate([(motor_height+prolonging+2*thickness)/2+wheel_thickness/2+2,0,0])
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))-(
    shank_length/sqrt(2))+2*connection_r])
wheel_f(fillet=fillet,wheel_thickness=wheel_thickness,wheel_r=wheel_r,connection_h=
    connection_h,connection_r=connection_r,screw_r=screw_r);}
if(display==1){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
motor_stand_f(display=1,cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=
    fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
color([0.5,0.5,0.5])
difference(){
rotate([-135,0,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)/2])
cube_f((motor_height+prolonging+2*thickness)/8,(motor_height+prolonging+2*thickness)/2,
    thigh_length,(motor_height+prolonging+2*thickness)/2);
translate([(motor_height+prolonging+2*thickness)/4,0,0])
rotate([0,90,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
cylinder((motor_height+prolonging+2*thickness),(sqrt(2)*motor_width+2*thickness)/2,(sqrt(2)*
    motor_width+2*thickness)/2);
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    translate([(motor_height+prolonging+2*thickness)/4,0,0])
    rotate([0,90,0])
    translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
    cylinder(h=(motor_height+prolonging+2*thickness),r=connection_r/2);
    rotate([-135,0,0])
    translate([-(motor_height+prolonging+2*thickness)/4,0,-((motor_height+prolonging+2*
        thickness)/4)/2])
    cube([(motor_height+prolonging+2*thickness),thigh_length-4*connection_r,(motor_height+
        prolonging+2*thickness)/4]);
    }
    }
if(display==2){
    prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
motor_stand_f(display=2,cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=
    fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
    }
if(display==3){
    prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
```

```
union(){
translate([0,0,-prolonging/2])
{
    ser_motor_f();}
}
    }
if(display==4){
    prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
rotate([0,90,0])
union(){
translate([0,0,-prolonging/2])
{
    motor_coupling(fillet =fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length,motor_axis_r=motor_axis_r,motor_axis_offset_r = motor_axis_offset_r
        ,motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);}
}
    }
if(display==5){
color([1,1,1])
union(){
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([-45,0,0])
union(){
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,6*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}
rotate([0,90,0])
cylinder_f(2*connection_r/10,2*connection_r,connection_r+thickness,connection_r+thickness);
}
translate([(motor_height+prolonging+2*thickness)/2+2,0,0])
rotate([0,-90,0])
male_connector(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r);
}
    }
if(display==6){
color([0.8,0.8,0.8])
translate([(motor_height+prolonging+2*thickness)/2-(connection_r),-(4*connection_r)/sqrt(2)
    ,(4*connection_r)/sqrt(2)])
translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt(2)])
rotate([45,0,0])
union(){
difference(){
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
translate([0,thigh_length-(2*connection_r),0])
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r);
    }
difference(){
rotate([0,90,0])
cylinder_f(1,3*connection_r,connection_r/2,connection_r/2);
rotate([0,90,0])
cylinder(3*connection_r,screw_r,screw_r);
    }
translate([0,0,-connection_r/2])
cube_f(connection_r/4,connection_r,thigh_length-(2*connection_r),connection_r);}
    }
if(display==7){
color([0.4,0.4,0.4])
difference(){
union(){
```

```
translate([(motor_height+prolonging+2*thickness)/2,-(thigh_length-2*connection_r)/sqrt(2),-(
    thigh_length-2*connection_r)/sqrt(2)])
rotate([-45,0,0])
difference(){
translate([0,-5*connection_r,-connection_r])
cube_f(2*connection_r/4,2*connection_r,shank_length+5*connection_r,2*connection_r);
translate([0,-(4*connection_r),0])
rotate([0,90,0])
translate([0,0,-connection_r])
cylinder(4*connection_r,connection_r/2,connection_r/2);
}}
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,0])
translate([0,0,-(thigh_length/sqrt(2))-(shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
translate([0,0,-((motor_height+prolonging+2*thickness)/2)])
cylinder((motor_height+prolonging+2*thickness),(sqrt(2)*motor_width+2*thickness)/2,(sqrt(2)*
    motor_width+2*thickness)/2);
}
color([0.4,0.4,0.4])
difference(){
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,0])
    cylinder_f(5,(motor_height+prolonging+2*thickness)/2+2*connection_r,connection_r/2,
        connection_r/2);
    translate([0,-(thigh_length-2*connection_r)/sqrt(2),-(thigh_length-2*connection_r)/sqrt
        (2)])
    rotate([0,90,0])
    translate([0,0,-1*connection_r])  cylinder((motor_height+prolonging+2*thickness)/2+4*
        connection_r,screw_r,screw_r);
    }
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))-(
    shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
union(){
motor_stand_f(display=1,cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=
    fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
}
    }
if(display==8){
    translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))
        -(shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
union(){
motor_stand_f(display=2,cube_shape=cube_shape,motor_axis_offset_r=motor_axis_offset_r,fillet=
    fillet,connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,motor_width=
    motor_width,motor_height=motor_height,motor_screw_hole_position=motor_screw_hole_position
    ,motor_axis_offset_h=motor_axis_offset_h,motor_axis_length=motor_axis_length,thickness=
    thickness);
}
    }
if (display==9){
    translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))
        -(shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
    }
if(display==10){
    translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))
        -(shank_length/sqrt(2))+2*connection_r])
rotate([0,90,0])
union(){
translate([0,0,-prolonging/2])
{
```

```
    motor_coupling(fillet =fillet,motor_height = motor_height,motor_axis_length=
        motor_axis_length,motor_axis_r=motor_axis_r,motor_axis_offset_r = motor_axis_offset_r
        ,motor_axis_offset_h = motor_axis_offset_h,connection_h=connection_h,connection_r=
        connection_r,thickness = thickness,screw_r = screw_r);}
}
    }
if(display==11){
translate([(motor_height+prolonging+2*thickness)/2+wheel_thickness/2+2,0,0])
translate([(motor_height+prolonging+2*thickness)/2+connection_r,0,-(thigh_length/sqrt(2))-(
    shank_length/sqrt(2))+2*connection_r])
wheel_f(fillet=fillet,wheel_thickness=wheel_thickness,wheel_r=wheel_r,connection_h=
    connection_h,connection_r=connection_r,screw_r=screw_r);}
if(display==1){
prolonging = motor_axis_offset_h + motor_axis_length + connection_h -2;
    }
}
```

c) : Mecanum-wheel.

```
//File name: mecanum_wheel.scad
//Purpose: generate a mecanum wheel.
//Parameter:
//1. display: display settings. when set to true, display the entire assembled module; when
    set to a specific number, the corresponding number of parts is displayed.
//2. fillet: the fillet radius.
//3. wheel_thickness: the thickness of the wheel.
//4. wheel_r: the radius of the wheel.
//5. connection_h: the length of the connector.
//6. connection_r: the circumradius of the connector.
//7. screw_r: the radius of the screw hole.
include<male_connector.scad>
include<cylinder_f.scad>
module roller_single(){
    color([235/255,46/255,46/255])
    import("roller.stl");
    }
module wheel_hub(connection_h,connection_r,screw_r,wheel_r=125,wheel_thickness){
    color([85/255,78/255,78/255])
    scale([wheel_r/27,wheel_r/27,wheel_r/27])
    import("wheel_hub.stl");
    color([85/255,78/255,78/255])
    rotate([0,90,0])
    translate([0,0,-wheel_thickness/2])
    scale([wheel_r/27,wheel_r/27,wheel_thickness/25])
    cylinder_f(5,25,15,15);
    color([85/255,78/255,78/255])
    rotate([0,-90,0])
    translate([0,0,wheel_thickness/2])
    male_connector(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r);
    }
module roller_pair_all(){
    for (i=[0:1:9]){
        rotate([36*i,0,0])
        union(){
        roller_single();
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();}
        }
    }
module roller_pair_1(){
        rotate([36*1,0,0])
        roller_single();
        rotate([36*0,0,0])
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();
    }
module roller_pair_2(){
```

```
        rotate([36*2,0,0])
        roller_single();
        rotate([36*1,0,0])
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();
    }
module roller_pair_3(){
        rotate([36*3,0,0])
        roller_single();
        rotate([36*2,0,0])
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();
    }
module roller_pair_4(){
        rotate([36*4,0,0])
        roller_single();
        rotate([36*3,0,0])
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();
    }
module roller_pair_5(){
        rotate([36*5,0,0])
        roller_single();
        rotate([36*4,0,0])
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();
    }
module roller_pair_6(){
        rotate([36*6,0,0])
        roller_single();
        rotate([36*5,0,0])
        mirror([0,0,1])
        mirror([1,0,0])
        roller_single();
    }
module roller_pair_7(){
    rotate([36*7,0,0])
    roller_single();
    rotate([36*6,0,0])
    mirror([0,0,1])
    mirror([1,0,0])
    roller_single();
}
module roller_pair_8(){
    rotate([36*8,0,0])
    roller_single();
    rotate([36*7,0,0])
    mirror([0,0,1])
    mirror([1,0,0])
    roller_single();
}
module roller_pair_9(){
    rotate([36*9,0,0])
    roller_single();
    rotate([36*8,0,0])
    mirror([0,0,1])
    mirror([1,0,0])
    roller_single();
}
module roller_pair_10(){
    rotate([36*10,0,0])
    roller_single();
    rotate([36*9,0,0])
    mirror([0,0,1])
```

```
    mirror([1,0,0])
    roller_single();
}
module mecanum_wheel(display=true,fillet,wheel_thickness,wheel_r=125,connection_h=20,
    connection_r=25,screw_r=3){
    union(){
    if (display == true)
        {
         scale([wheel_r/27,wheel_r/27,wheel_r/27])
         roller_pair_all();
         wheel_hub(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,
             wheel_r=wheel_r,wheel_thickness=wheel_thickness);
        }
    if (display == 1)
        {
         scale([wheel_r/27,wheel_r/27,wheel_r/27])
         roller_pair_1();}
    if (display == 2)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_2();}
    if (display == 3)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_3();}
    if (display == 4)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_4();}
    if (display == 5)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_5();}
    if (display == 6)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_6();}
    if (display == 7)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_7();}
    if (display == 8)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_8();}
    if (display == 9)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_9();}
    if (display == 10)
        {scale([wheel_r/27,wheel_r/27,wheel_r/27])
            roller_pair_10();}
    if (display == 11)
    {wheel_hub(connection_h=connection_h,connection_r=connection_r,screw_r=screw_r,wheel_r=
        wheel_r,wheel_thickness=wheel_thickness);}
    }
}
```

II. PROMPT ENGINEERING

In ChatBuilder, there are two agents: Planner and Assembler, each with a prompt consisting of four parts: General Description, Subtask Description, Input Example, and Output Example. The specific prompts are as follows:

A. *Planner*

a) : General Description.

```
Based on the input text, program and model the robot structure using OpenSCAD.
The modeled robot structure is composed of the smallest units assembled in different ways.
```

b) : Subtask Description.

```
This process is divided into two steps:
Step 1: Analyze the input text to determine the construction information of the smallest
    units and the required parts.
Step 2: Analyze the characteristics of the parts to determine the assembly positions.
You only need to be responsible for the first step.
```

c) : Input Example I.

```
Below is a case example,for robotic arm, including an input and an output.
Note that your output must strictly follow the format of the case and strictly adhere to the
    rules in the case comments:
Input:
Build a structure for a robot arm with six degrees of freedom.I prefer the entire robot arm
    to have a cylindrical shape.
```

d) : Output Example I.

```
Output: (Note, pay attention to the key information in the comments, imitate this reasoning)
//Input text: Build a structure for a robot arm with six degrees of freedom.I prefer the
    entire robot arm to have a cylindrical shape.
//Unit size
//Since the input text mentions preferring a cylindrical shape for the robot arm,
//set cube_shape to false and calculate container_width as container_width=sqrt(2)*
    motor_width,
//Other information is not given, so set to default values
fillet=2;
motor_axis_offset_r=35;//The radius of the circular area beneath the motor shaft, which is
    equal to the radius of the coupling.
motor_axis_offset_h=3;//The thickness of the circular area beneath the motor shaft.

motor_axis_length=32;//the length of the motor shaft
motor_axis_r=9.5;//the radius of the motor shaft

motor_width=80;
motor_height=123;

screw_r=3;//the radius of the screw hole
motor_screw_hole_position=8.2;//the position of the motor screw holes

thigh_length = 350;//the thigh length of a leg or leg_wheel
shank_length = 350;//the shank length of a leg or leg_wheel

wheel_thickness=90;//The wheel width of a wheeled robot
wheel_r=125;//The wheel radius of a wheeled robot
wheel_track=400;//The distance between the centers of two wheels on the same axis of a
    wheeled robot
chassis_length=200;//Wheeled robot chassis length
chassis_thickness=150;//the thickness of wheeled robot chassis

connection_h=20;//the length of the connecting shaft.
connection_r=25;//the radius of the connecting shaft.
thickness=8; // The larger it is, the greater the thickness of the part, which makes the
    robot stiffer or sturdier
cube_shape=false;

container_width=sqrt(2)*motor_width-connection_h+2; // If cube shape is false, this formula
    is used
//container_width=(screw_r+3)*4+motor_width-connection_h+2; //If cube shape is true, this
    formula is used

//Part plan
//The following rules must be observed:
//If the current part is joint_positive_z or link_positive_z, the next part must not be
    joint_negative_z or link_negative_z
//If the current part is joint_negative_z or link_negative_z, the next part must not be
    joint_positive_z or link_positive_z
//The last part must be a link type, and the penultimate part must be a joint type

//Since the input text is to build a six-degree-of-freedom robot arm structure, six joint
    type parts are required

//part_1: choose joint_positive_z as joint_1: base_joint or waist_joint,
//This is the first degree of freedom
part_1_w = joint_positive_z_w;
part_1_h = joint_positive_z_h;
```

```
//part_2: choose link_positive_z as link_between_joint_1_and_joint_2,
//This part can be omitted
//The height of the base can be increased by increasing the number of link parts, They are
    numbered part_2_1, part_2_2, etc
part_2_1_w = link_positive_z_w;
part_2_1_h = link_positive_z_h;

//part_3: choose joint_negative_x as joint_2: shoulder_joint,
//This is the second degree of freedom
part_3_w = joint_negative_x_w;
part_3_h = joint_negative_x_h;

//part_4: two link_positive_z together form link_between_joint_2_and_joint_3: upper_arm,
//upper_arm must contain at least two links
//This part is necessary and can be extended, They are numbered part_4_1, part_4_2, etc
part_4_1_w = link_positive_z_w;
part_4_1_h = link_positive_z_h;
part_4_2_w = link_positive_z_w;
part_4_2_h = link_positive_z_h;

//part_5: choose joint_positive_x as joint_3: elbow_joint,
//This is the third degree of freedom
part_5_w = joint_positive_x_w;
part_5_h = joint_positive_x_h;

//part_6: two link_positive_z together form link_between_joint_3_and_joint_4: lower_arm or
    forearm,
//lower_arm or forearm must contain at least two links
//This part is necessary and can be extended, They are numbered part_6_1, part_6_2, etc
part_6_1_w = link_positive_z_w;
part_6_1_h = link_positive_z_h;
part_6_2_w = link_positive_z_w;
part_6_2_h = link_positive_z_h;

//part_7, part_8, part_9: three joint type parts form three wrist_joints, corresponding to
    the fourth, fifth, and sixth degrees of freedom
//If the robot arm structure mentions wrist_joints, at least wrist_joint_1 is required
//part_7: choose joint_positive_z as joint_4: wrist_joint_1,
//This is the fourth degree of freedom
part_7_w = joint_positive_z_w;
part_7_h = joint_positive_z_h;

//part_8: choose joint_negative_x as joint_5: wrist_joint_2,
//This is the fifth degree of freedom
part_8_w = joint_negative_x_w;
part_8_h = joint_negative_x_h;

//part_9: choose joint_positive_z as joint_6: wrist_joint_3,
//This is the sixth degree of freedom
part_9_w = joint_positive_z_w;
part_9_h = joint_positive_z_h;

//part_10: choose link_positive_z as the last part: end_effector,
//Cannot be omitted, usually choose link_positive_z as the last part
part_10_w = link_positive_z_w;
part_10_h = link_positive_z_h;

////The following code is only used when you need a gripper or soft_gripper to be the
    end_effector
//part_11_w = gripper_end_w;//part_11_w = soft_gripper_end_w;
//part_11_h = gripper_end_h;//part_11_h = soft_gripper_end_h;

////The following code is only used if the arm is on the body or if there is a chassis as a
    base under the arm
//part_0_w = chassis_w;
//part_0_h = chassis_h;
//part_0_l = chassis_l;
```

e) : Input Example II.

```
Below is another case example, for wheeled robot, including an input and an output.
Note that your output must strictly follow the format of the case and strictly adhere to the
    rules in the case comments:
Input:
Build a structure for a wheeled robot with four wheels.The two front wheels are steering
    wheels. The two rear wheels are fixed wheels.
```

f) : Output Example II.

```
Output: (Note, pay attention to the key information in the comments, imitate this reasoning)
//Input text: Build a structure for a wheeled robot with four wheels.The two front wheels are
     steering wheels. The two rear wheels are fixed wheels.
//Unit size
//set cube_shape to false and calculate container_width as container_width=sqrt(2)*
    motor_width,
//Other information is not given, so set to default values
fillet=2;
motor_axis_offset_r=35;//The radius of the circular area beneath the motor shaft, which is
    equal to the radius of the coupling.
motor_axis_offset_h=3;//The thickness of the circular area beneath the motor shaft.

motor_axis_length=32;//the length of the motor shaft
motor_axis_r=9.5;//the radius of the motor shaft

motor_width=80;
motor_height=123;

screw_r=3;//the radius of the screw hole
motor_screw_hole_position=8.2;//the position of the motor screw holes

thigh_length = 350;//the thigh length of a leg or leg_wheel
shank_length = 350;//the shank length of a leg or leg_wheel

wheel_thickness=90;//The wheel width of a wheeled robot
wheel_r=125;//The wheel radius of a wheeled robot
wheel_track=400;//The distance between the centers of two wheels on the same axis of a
    wheeled robot
chassis_length=200;//Wheeled robot chassis length
chassis_thickness=150;//the thickness of wheeled robot chassis

connection_h=20;//the length of the connecting shaft.
connection_r=25;//the radius of the connecting shaft.
thickness=8; // The larger it is, the greater the thickness of the part, which makes the
    robot stiffer or sturdier
cube_shape=false;

container_width=sqrt(2)*motor_width-connection_h+2; // If cube shape is false, this formula
    is used
//container_width=(screw_r+3)*4+motor_width-connection_h+2; //If cube shape is true, this
    formula is used

//Part plan
//The following rules must be observed:
//If the current part is of the chassis type, you must provide its w, h, and l parameters. If
     it is not, you only need to provide w and h.
//Since the input text is to build a four-wheeled robot structure, four wheels are required.
//It is common to start planning parts from the rear wheels.
//part_1:choose chassis as the axle connecting the two rear wheels,two rear legs or two rear
    leg_wheels
part_1_w = chassis_w;
part_1_h = chassis_h;
part_1_l = chassis_l;

//part_2:The right (rear) fixed wheel is composed of four parts, which are:link_positive_x,
    link_negative_z,joint_positive_x,wheel_right or mecanum_wheel_right
part_2_1_w = link_positive_x_w;
part_2_1_h = link_positive_x_h;
```

```
part_2_2_w = link_negative_z_w;
part_2_2_h = link_negative_z_h;

part_2_3_w = joint_positive_x_w;
part_2_3_h = joint_positive_x_h;

part_2_4_w = wheel_right_w;//part_2_4_w = mecanum_wheel_rear_right_w;
part_2_4_h = wheel_right_h;//part_2_4_h = mecanum_wheel_rear_right_h;

////part_2:The right leg or leg_wheel is composed of two parts, which are:joint_positive_x,
    leg_right or leg_wheel_right
//part_2_1_w = joint_positive_x_w;
//part_2_1_h = joint_positive_x_h;

//part_2_2_w = leg_right_w;//part_2_2_w = leg_wheel_right_w;
//part_2_2_h = leg_right_h;//part_2_2_h = leg_wheel_right_h;

//part_3:The left (rear) fixed wheel is composed of four parts, which are:link_negative_x,
    link_negative_z,joint_negative_x,wheel_left or mecanum_wheel_left
part_3_1_w = link_negative_x_w;
part_3_1_h = link_negative_x_h;

part_3_2_w = link_negative_z_w;
part_3_2_h = link_negative_z_h;

part_3_3_w = joint_negative_x_w;
part_3_3_h = joint_negative_x_h;

part_3_4_w = wheel_left_w;//part_3_4_w = mecanum_wheel_rear_left_w;
part_3_4_h = wheel_left_h;//part_3_4_h = mecanum_wheel_rear_left_h;

////part_3:The left leg or leg_wheel is composed of two parts, which are:joint_positive_x,
    leg_left or leg_wheel_left
//part_3_1_w = joint_positive_x_w;
//part_3_1_h = joint_positive_x_h;

//part_3_2_w = leg_left_w;//part_3_2_w = leg_wheel_left_w;
//part_3_2_h = leg_left_h;//part_3_2_h = leg_wheel_left_h;

//part_4:choose chassis-type parts for the body, you can extend the body length or increase
    the wheelbase by adding more chassis parts
part_4_w = chassis_w;
part_4_h = chassis_h;
part_4_l = chassis_l;

//part_5:choose chassis as the axle connecting the two front wheels,two front legs or two
    front leg_wheels
part_5_w = chassis_w;
part_5_h = chassis_h;
part_5_l = chassis_l;

//part_6:The right (front) steering wheel is composed of four parts, which are:
    link_positive_x,joint_negative_z,joint_positive_x,wheel_right or mecanum_wheel_right
part_6_1_w = link_positive_x_w;
part_6_1_h = link_positive_x_h;

part_6_2_w = joint_negative_z_w;
part_6_2_h = joint_negative_z_h;

part_6_3_w = joint_positive_x_w;
part_6_3_h = joint_positive_x_h;

part_6_4_w = wheel_right_w;//part_6_4_w = mecanum_wheel_front_right_w;
part_6_4_h = wheel_right_h;//part_6_4_h = mecanum_wheel_front_right_h;

////When you need to build a legged robot or leg-wheeled robot, use the following code to
    build leg or leg_wheel
```

```
////part_6:The right leg or leg_wheel is composed of two parts, which are:joint_positive_x,
    leg_right or leg_wheel_right
//part_6_1_w = joint_positive_x_w;
//part_6_1_h = joint_positive_x_h;

//part_6_2_w = leg_right_w;//part_6_2_w = leg_wheel_right_w;
//part_6_2_h = leg_right_h;//part_6_2_h = leg_wheel_right_h;

//part_7:The left (front) steering wheel is composed of four parts, which are:link_negative_x
    ,joint_negative_z,joint_negative_x,wheel_left or mecanum_wheel_left
part_7_1_w = link_negative_x_w;
part_7_1_h = link_negative_x_h;

part_7_2_w = joint_negative_z_w;
part_7_2_h = joint_negative_z_h;

part_7_3_w = joint_negative_x_w;
part_7_3_h = joint_negative_x_h;

part_7_4_w = wheel_left_w;//part_7_4_w = mecanum_wheel_front_left_w;
part_7_4_h = wheel_left_h;//part_7_4_h = mecanum_wheel_front_left_h;

////When you need to build a legged robot or leg-wheeled robot, use the following code to
    build leg or leg_wheel
////part_7:The left leg or leg_wheel is composed of two parts, which are:joint_positive_x,
    leg_left or leg_wheel_left
//part_7_1_w = joint_positive_x_w;
//part_7_1_h = joint_positive_x_h;

//part_7_2_w = leg_left_w;//part_7_2_w = leg_wheel_left_w;
//part_7_2_h = leg_left_h;//part_7_2_h = leg_wheel_left_h;

Now I will start the input, please complete your task according to the above requirements.
```

B. Assembler

a) : General Description.

```
Based on the input text, program and model the robot structure using OpenSCAD.
The modeled robot structure is composed of the smallest units assembled in different ways.
```

b) : Subtask Description.

```
This process is divided into two steps:
Step 1: Analyze the input text to determine the construction information of the smallest
    units and the required parts.
Step 2: Analyze the characteristics of the parts to determine the assembly positions.
You only need to be responsible for the second step.
```

c) : Input Example I.

```
Below is a case example, for robotic arm including an input and an output.
Note that your output must strictly follow the format of the case and strictly adhere to the
    rules in the case comments:
Input:
//Input text: Build a structure for a robot arm with six degrees of freedom.I prefer the
    entire robot arm to have a cylindrical shape.
//Unit size
fillet=2;
motor_axis_offset_r=35;
motor_axis_offset_h=3;
motor_axis_length=32;
motor_axis_r=9.5;
motor_width=80;
motor_height=123;
screw_r=3;
motor_screw_hole_position=8.2;
thigh_length = 350;
shank_length = 350;
wheel_thickness=90;
```

```
wheel_r=125;
wheel_track=400;
chassis_length=200;
chassis_thickness=150;
connection_h=20;
connection_r=25;
thickness=8;
cube_shape=false;
container_width=sqrt(2)*motor_width-connection_h+2;
//container_width=(screw_r+3)*4+motor_width-connection_h+2;

//Part plan
//part_1: choose joint_positive_z as joint_1: base_joint or waist_joint,
//This is the first degree of freedom
part_1_w = joint_positive_z_w;
part_1_h = joint_positive_z_h;

//part_2: choose link_positive_z as link_between_joint_1_and_joint_2,
part_2_1_w = link_positive_z_w;
part_2_1_h = link_positive_z_h;

//part_3: choose joint_negative_x as joint_2: shoulder_joint,
part_3_w = joint_negative_x_w;
part_3_h = joint_negative_x_h;

//part_4: two link_positive_z together form link_between_joint_2_and_joint_3: upper_arm,
part_4_1_w = link_positive_z_w;
part_4_1_h = link_positive_z_h;
part_4_2_w = link_positive_z_w;
part_4_2_h = link_positive_z_h;

//part_5: choose joint_positive_x as joint_3: elbow_joint,
part_5_w = joint_positive_x_w;
part_5_h = joint_positive_x_h;

//part_6: two link_positive_z together form link_between_joint_3_and_joint_4: lower_arm or
    forearm,
part_6_1_w = link_positive_z_w;
part_6_1_h = link_positive_z_h;
part_6_2_w = link_positive_z_w;
part_6_2_h = link_positive_z_h;

//part_7, part_8, part_9: three joint type parts form three wrist_joints, corresponding to
    the fourth, fifth, and sixth degrees of freedom
//part_7: choose joint_positive_z as joint_4: wrist_joint_1,
//This is the fourth degree of freedom
part_7_w = joint_positive_z_w;
part_7_h = joint_positive_z_h;

//part_8: choose joint_negative_x as joint_5: wrist_joint_2,
part_8_w = joint_negative_x_w;
part_8_h = joint_negative_x_h;

//part_9: choose joint_positive_z as joint_6: wrist_joint_3,
part_9_w = joint_positive_z_w;
part_9_h = joint_positive_z_h;

//part_10: choose link_positive_z as the last part: end_effector,
part_10_w = link_positive_z_w;
part_10_h = link_positive_z_h;

////The following code is only used if the arm is on the body or if there is a chassis as a
    base under the arm
//part_0_w = chassis_w;
//part_0_h = chassis_h;
//part_0_l = chassis_l;
```

d) : Output Example I.

Output: (Note, pay attention to the key information in the comments, imitate this reasoning)

```
// Assembly position coordinate
// The following rules must be observed:
// From part_2 onwards, before calculating the coordinates of the current part, do the following:
// Check if the previous part is of joint type,
// If the previous part is a joint type, the current part has an offset in the direction of the previous joint part,
// For example, the previous part of part_2 (part_1) is joint_positive_z, so part_2 has a positive offset in the z-axis direction (positive_z)
// If the previous part is not a joint type, meaning the previous part is a link type, check if the current part is also a link type,
// If the current part is a link type, then the current part has an offset in the direction of the previous link part,
// For example, part_5 itself is a link type, and its previous part is link_positive_z, both being link types,
// so part_5 has a positive offset in the z-axis direction (positive_z)
// In other cases, the current part generally only has a positive offset in the z-axis direction

////The following code is only used if the arm is on the body or if there is a chassis as a base under the arm
//part_0_x = 0;
//part_0_y = 0;
//part_0_z = 0;
//part_1_x = part_0_x;
//part_1_y = part_0_y;
//part_1_z = part_0_z + part_0_h/2 + part_1_h/2;

// part_1: choose joint_positive_z as joint_1: base_joint or waist_joint,
// The coordinates of part_1 are both 0
part_1_x = 0;
part_1_y = 0;
part_1_z = 0;

// part_2: choose link_positive_z as link_between_joint_1_and_joint_2,
// As the previous part is joint_positive_z, part_2_1 has a positive offset in the z-axis direction (positive_z)
part_2_1_x = part_1_x;
part_2_1_y = part_1_y;
part_2_1_z = part_1_z + (part_1_h/2 + part_2_1_h/2);

// part_3: choose joint_negative_x as joint_2: shoulder_joint,
// As the previous part is not of joint type, and the current part is not of link type, so there is only a positive offset in the z-axis
part_3_x = part_2_1_x;
part_3_y = part_2_1_y;
part_3_z = part_2_1_z + (part_2_1_h/2 + part_3_h/2);

// part_4: two link_positive_z together form link_between_joint_2_and_joint_3: upper_arm
// part_4_1: link_positive_z
// As the previous part is joint_negative_x, part_4_1 has a negative offset in the x-axis direction (negative_x)
part_4_1_x = part_3_x - (part_3_w/2 + part_4_1_w/2);
part_4_1_y = part_3_y;
part_4_1_z = part_3_z;

// part_4_2: link_positive_z
// As the previous part is not of joint type, but the current part is a link type,
// and the previous part is link_positive_z, so part_4_2 has a positive offset in the z-axis direction (positive_z)
part_4_2_x = part_4_1_x;
part_4_2_y = part_4_1_y;
part_4_2_z = part_4_1_z + (part_4_1_h/2 + part_4_2_h/2);

// part_5: choose joint_positive_x as joint_3: elbow_joint,
// As the previous part is not of joint type, and the current part is not of link type, so there is only a positive offset in the z-axis
```

```
part_5_x = part_4_2_x;
part_5_y = part_4_2_y;
part_5_z = part_4_2_z + (part_4_2_h/2 + part_5_h/2);

// part_6: two link_positive_z together form link_between_joint_3_and_joint_4: lower_arm or
    forearm
// part_6_1: link_positive_z
// As the previous part is joint_positive_x, part_6_1 has a positive offset in the x-axis
    direction (positive_x)
part_6_1_x = part_5_x + (part_5_w/2 + part_6_1_w/2);
part_6_1_y = part_5_y;
part_6_1_z = part_5_z;

// part_6_2: link_positive_z
// As the previous part is not of joint type, , but the current part is a link type,
// and the previous part is link_positive_z, part_6_2 has a positive offset in the z-axis
    direction (positive_z)
part_6_2_x = part_6_1_x;
part_6_2_y = part_6_1_y;
part_6_2_z = part_6_1_z + (part_6_1_h/2 + part_6_2_h/2);

// part_7, part_8, part_9: three joint type parts form three wrist_joints, corresponding to
    the fourth, fifth, and sixth degrees of freedom
// part_7: choose joint_positive_z as joint_4: wrist_joint_1,
// As the previous part is not of joint type, and the current part is not of link type, so
    there is only a positive offset in the z-axis
part_7_x = part_6_2_x;
part_7_y = part_6_2_y;
part_7_z = part_6_2_z + (part_6_2_h/2 + part_7_h/2);

// part_8: choose joint_negative_x as joint_5: wrist_joint_2,
// As the previous part is joint_positive_z, part_8 has a positive offset in the z-axis
    direction (positive_z)
part_8_x = part_7_x;
part_8_y = part_7_y;
part_8_z = part_7_z + (part_7_h/2 + part_8_h/2);

// part_9: choose joint_positive_z as joint_6: wrist_joint_3,
// As the previous part is joint_negative_x, part_9 has a negative offset in the x-axis
    direction (negative_x)
part_9_x = part_8_x - (part_8_w/2 + part_9_w/2);
part_9_y = part_8_y;
part_9_z = part_8_z;

// part_10: choose link_positive_z as the last part: end_effector,
// As the previous part is joint_positive_z, part_10 has a positive offset in the z-axis
    direction (positive_z)
part_10_x = part_9_x;
part_10_y = part_9_y;
part_10_z = part_9_z + (part_9_h/2 + part_10_h/2);

//// The following code is only used when you need a gripper or soft_gripper to be the
    end_effector
//// part_11: gripper or soft_gripper
//part_11_x = part_10_x;
//part_11_y = part_10_y;
//part_11_z = part_10_z + (part_10_h/2 + part_11_h/2);

// Move the corresponding parts according to the assembly coordinates
// The following symbol must be present.
/*check*/
// part_1: choose joint_positive_z as joint_1: base_joint or waist_joint,
translate([part_1_x, part_1_y, part_1_z])
joint_positive_z(exploded=exploded);

/*check*/
// part_2_1: choose link_positive_z as link_between_joint_1_and_joint_2,
translate([part_2_1_x, part_2_1_y, part_2_1_z])
```

```
link_positive_z(exploded=exploded);

/*check*/
// part_3: choose joint_negative_x as joint_2: shoulder_joint,
translate([part_3_x, part_3_y, part_3_z])
joint_negative_x(exploded=exploded);

/*check*/
// part_4: two link_positive_z together form link_between_joint_2_and_joint_3: upper_arm
// part_4_1: link_positive_z
translate([part_4_1_x, part_4_1_y, part_4_1_z])
link_positive_z(exploded=exploded);

/*check*/
// part_4_2: link_positive_z
translate([part_4_2_x, part_4_2_y, part_4_2_z])
link_positive_z(exploded=exploded);

/*check*/
// part_5: choose joint_positive_x as joint_3: elbow_joint,
translate([part_5_x, part_5_y, part_5_z])
joint_positive_x(exploded=exploded);

/*check*/
// part_6: two link_positive_z together form link_between_joint_3_and_joint_4: lower_arm or
    forearm
// part_6_1: link_positive_z
translate([part_6_1_x, part_6_1_y, part_6_1_z])
link_positive_z(exploded=exploded);

/*check*/
// part_6_2: link_positive_z
translate([part_6_2_x, part_6_2_y, part_6_2_z])
link_positive_z(exploded=exploded);

/*check*/
// part_7, part_8, part_9: three joint type parts form three wrist_joints,
// part_7: choose joint_positive_z as joint_4: wrist_joint_1,
translate([part_7_x, part_7_y, part_7_z])
joint_positive_z(exploded=exploded);

/*check*/
// part_8: choose joint_negative_x as joint_5: wrist_joint_2,
translate([part_8_x, part_8_y, part_8_z])
joint_negative_x(exploded=exploded);

/*check*/
// part_9: choose joint_positive_z as joint_6: wrist_joint_3,
translate([part_9_x, part_9_y, part_9_z])
joint_positive_z(exploded=exploded);

/*check*/
// part_10: choose link_positive_z as the last part: end_effector,
translate([part_10_x, part_10_y, part_10_z])
link_positive_z(exploded=exploded);

//// /*check*/
//// The following code is only used when you need a gripper or soft_gripper to be the
    end_effector
//// part_11:gripper_end() or soft_gripper_end()
//translate([part_11_x, part_11_y, part_11_z])
//gripper_end(exploded=exploded);//soft_gripper_end(exploded=exploded);
```

e) : Input Example II.

```
Below is another case example, for wheeled robot, including an input and an output.
Note that your output must strictly follow the format of the case and strictly adhere to the
    rules in the case comments:
Input:
```

```
//Input text: Build a structure for a wheeled robot with four wheels.The two front wheels are
     steering wheels. The two rear wheels are fixed wheels.
//Unit size
fillet=2;
motor_axis_offset_r=35;
motor_axis_offset_h=3;
motor_axis_length=32;
motor_axis_r=9.5;
motor_width=80;
motor_height=123;
screw_r=3;
motor_screw_hole_position=8.2;
thigh_length = 350;
shank_length = 350;
wheel_thickness=90;
wheel_r=125;
wheel_track=400;
chassis_length=200;
chassis_thickness=150;
connection_h=20;
connection_r=25;
thickness=8;
cube_shape=false;
container_width=sqrt(2)*motor_width-connection_h+2;
//container_width=(screw_r+3)*4+motor_width-connection_h+2;

//Part plan
part_1_w = chassis_w;
part_1_h = chassis_h;
part_1_l = chassis_l;

//part_2:The right (rear) fixed wheel is composed of four parts, which are:link_positive_x,
    link_negative_z,joint_positive_x,wheel_right
part_2_1_w = link_positive_x_w;
part_2_1_h = link_positive_x_h;

part_2_2_w = link_negative_z_w;
part_2_2_h = link_negative_z_h;

part_2_3_w = joint_positive_x_w;
part_2_3_h = joint_positive_x_h;

part_2_4_w = wheel_right_w;
part_2_4_h = wheel_right_h;

//part_3:The left (rear) fixed wheel is composed of four parts, which are:link_negative_x,
    link_negative_z,joint_negative_x,wheel_left
part_3_1_w = link_negative_x_w;
part_3_1_h = link_negative_x_h;

part_3_2_w = link_negative_z_w;
part_3_2_h = link_negative_z_h;

part_3_3_w = joint_negative_x_w;
part_3_3_h = joint_negative_x_h;

part_3_4_w = wheel_left_w;
part_3_4_h = wheel_left_h;

//part_4:choose chassis-type parts for the body, you can extend the body length or increase
    the wheelbase by adding more chassis parts
part_4_w = chassis_w;
part_4_h = chassis_h;
part_4_l = chassis_l;

//part_5:choose chassis as the axle connecting the two front wheels
part_5_w = chassis_w;
part_5_h = chassis_h;
```

```
part_5_l = chassis_l;

//part_6:The right (front) steering wheel is composed of four parts, which are:
    link_positive_x,joint_negative_z,joint_positive_x,wheel_right
part_6_1_w = link_positive_x_w;
part_6_1_h = link_positive_x_h;

part_6_2_w = joint_negative_z_w;
part_6_2_h = joint_negative_z_h;

part_6_3_w = joint_positive_x_w;
part_6_3_h = joint_positive_x_h;

part_6_4_w = wheel_right_w;
part_6_4_h = wheel_right_h;

//part_7:The left (front) steering wheel is composed of four parts, which are:link_negative_x
    ,joint_negative_z,joint_negative_x,wheel_left
part_7_1_w = link_negative_x_w;
part_7_1_h = link_negative_x_h;

part_7_2_w = joint_negative_z_w;
part_7_2_h = joint_negative_z_h;

part_7_3_w = joint_negative_x_w;
part_7_3_h = joint_negative_x_h;

part_7_4_w = wheel_left_w;
part_7_4_h = wheel_left_h;
```

f) : Output Example II.

```
Output: (Note, pay attention to the key information in the comments, imitate this reasoning)
//Assembly position coordinate
//The following rules must be observed:
//Chassis-type parts must specify coordinates in the x, y, and z dimensions,
//Except for the rear wheels (the first pair of wheels), which have no offset in the y
    direction,
//all other parts have offsets in the y direction.
//The y-direction offset for the same pair of wheels is the same,
//meaning it equals the y value of the previous part, with no additional offset.
//part_1:choose chassis as the axle connecting the two rear wheels
part_1_x = 0;
part_1_y = 0;
part_1_z = 0;

// part2: the right (rear) wheel, consists of four parts, which are: link_positive_x,
    link_negative_z, joint_positive_x, wheel_right
// part_2_1: link_positive_x
part_2_1_x = part_1_x + part_1_w/2 + part_2_1_w/2;
part_2_1_y = part_1_y;
part_2_1_z = part_1_z;

// part_2_2: link_negative_z
// Note that the second part of all wheels has an offset only in the x direction (positive
    for right, negative for left)
// and no offset in other directions.
part_2_2_x = part_2_1_x + part_2_1_w/2 + part_2_2_w/2;
part_2_2_y = part_2_1_y;
part_2_2_z = part_2_1_z;

// part_2_3: joint_positive_x
part_2_3_x = part_2_2_x;
part_2_3_y = part_2_2_y;
part_2_3_z = part_2_2_z - part_2_2_h/2 - part_2_3_h/2;

// part_2_4: wheel_right or mecanum_wheel_right
// Note that wheel_right or mecanum_wheel_right has a positive offset only in the x-axis
    direction and none in other direction.
```

```
part_2_4_x = part_2_3_x + part_2_3_w/2 + part_2_4_w/2;
part_2_4_y = part_2_3_y;
part_2_4_z = part_2_3_z;

////When you need to build a legged robot or leg-wheeled robot, use the following code
////part_2:The right leg or leg_wheel is composed of two parts, which are:joint_positive_x,
    leg_right or leg_wheel_right
////part_2_1:joint_positive_x
//part_2_1_x = part_1_x + part_1_w/2 + part_2_1_w/2;
//part_2_1_y = part_1_y;
//part_2_1_z = part_1_z;

////part_2_2:leg_right or leg_wheel_right
//part_2_2_x = part_2_1_x + part_2_1_w/2 + part_2_2_w/2;
//part_2_2_y = part_2_1_y;
//part_2_2_z = part_2_1_z;

// part3: the left (rear) wheel, consists of four parts
// part_3_1: link_negative_x
part_3_1_x = part_1_x - part_1_w/2 - part_3_1_w/2;
part_3_1_y = part_1_y;
part_3_1_z = part_1_z;

// part_3_2: link_negative_z
// Note that the second part of all wheels has an offset only in the x direction (positive
    for right, negative for left)
// and no offset in other directions.
part_3_2_x = part_3_1_x - part_3_1_w/2 - part_3_2_w/2;
part_3_2_y = part_3_1_y;
part_3_2_z = part_3_1_z;

// part_3_3: joint_negative_x
part_3_3_x = part_3_2_x;
part_3_3_y = part_3_2_y;
part_3_3_z = part_3_2_z - part_3_2_h/2 - part_3_3_h/2;

// part_3_4: wheel_left or mecanum_wheel_left
// Note that wheel_left or mecanum_wheel_left has a negative offset only in the x-axis
    direction and none in other direction.
part_3_4_x = part_3_3_x - part_3_3_w/2 - part_3_4_w/2;
part_3_4_y = part_3_3_y;
part_3_4_z = part_3_3_z;

////When you need to build a legged robot or leg-wheeled robot, use the following code
////part_3:The left leg or leg_wheel is composed of two parts, which are:joint_negative_x,
    leg_left or leg_wheel_left
////part_3_1:joint_negative_x
//part_3_1_x = part_1_x - part_1_w/2 - part_2_1_w/2;
//part_3_1_y = part_1_y;
//part_3_1_z = part_1_z;

////part_3_2:leg_left or leg_wheel_left
//part_3_2_x = part_3_1_x - part_3_1_w/2 - part_3_2_w/2;
//part_3_2_y = part_3_1_y;
//part_3_2_z = part_3_1_z;

//part_4:choose chassis-type parts for the body, you can extend the body length or increase
    the wheelbase by adding more chassis parts
//Generally, chassis-type parts used as the body only have offsets in the y direction.
part_4_x = part_1_x;
part_4_y = part_1_y + part_1_l/2 + part_4_l/2;
part_4_z = part_1_z;

//part_5:choose chassis as the axle connecting the two front wheels
part_5_x = part_4_x;
part_5_y = part_4_y + part_4_l/2 + part_5_l/2;
part_5_z = part_4_z;
```

```
//The y-direction offset for the same pair of wheels is the same,
//meaning it equals the y value of the previous part, with no additional offset.
//part_6:the right (front) wheel,consists of four parts
//part_6_1 : link_positive_x
part_6_1_x = part_5_x + part_5_w/2 + part_6_1_w/2;
part_6_1_y = part_5_y;
part_6_1_z = part_5_z;

//part_6_2: joint_negative_z
//Note that the second part of all wheels has an offset only in the x direction (positive for
     right, negative for left)
//and no offset in other directions.
part_6_2_x = part_6_1_x + part_6_1_w/2 + part_6_2_w/2;
part_6_2_y = part_6_1_y;
part_6_2_z = part_6_1_z;

//part_6_3:joint_positive_x
part_6_3_x = part_6_2_x;
part_6_3_y = part_6_2_y;
part_6_3_z = part_6_2_z - part_6_2_h/2 - part_6_3_h/2;

//part_6_4: wheel_right or mecanum_wheel_right
//Note that wheel_right or mecanum_wheel_right has a positive offset only in the x-axis
    direction and none in other direction.
part_6_4_x = part_6_3_x + part_6_3_w/2 + part_6_4_w/2;
part_6_4_y = part_6_3_y;
part_6_4_z = part_6_3_z;

//part_7:the left (front) wheel,consists of four parts
//part_7_1: link_negative_x
part_7_1_x = part_5_x - part_5_w/2 - part_7_1_w/2;
part_7_1_y = part_5_y;
part_7_1_z = part_5_z;

//part_7_2: joint_negative_z
//Note that the second part of all wheels has an offset only in the x direction (positive for
     right, negative for left)
//and no offset in other directions.
part_7_2_x = part_7_1_x - part_7_1_w/2 - part_7_2_w/2;
part_7_2_y = part_7_1_y;
part_7_2_z = part_7_1_z;

//part_7_3: joint_negative_x
part_7_3_x = part_7_2_x;
part_7_3_y = part_7_2_y;
part_7_3_z = part_7_2_z - part_7_2_h/2 - part_7_3_h/2;

//part_7_4: wheel_left or mecanum_wheel_left
//Note that wheel_left or mecanum_wheel_left has a negative offset only in the x-axis
    direction and none in other direction.
part_7_4_x = part_7_3_x - part_7_3_w/2 - part_7_4_w/2;
part_7_4_y = part_7_3_y;
part_7_4_z = part_7_3_z;

// Move the corresponding parts according to the assembly coordinates
//The following symbol must be present.
/*check*/
// part_1:chassis
translate([part_1_x,part_1_y,part_1_z])
chassis(exploded=exploded);

/*check*/
// part_2_1: link_positive_x
translate([part_2_1_x, part_2_1_y, part_2_1_z])
link_positive_x(exploded=exploded);

/*check*/
// part_2_2: link_negative_z
```

```
translate([part_2_2_x, part_2_2_y, part_2_2_z])
link_negative_z(exploded=exploded);

/*check*/
// part_2_3: joint_positive_x
translate([part_2_3_x, part_2_3_y, part_2_3_z])
joint_positive_x(exploded=exploded);

/*check*/
// part_2_4: wheel_right
translate([part_2_4_x, part_2_4_y, part_2_4_z])
wheel_right(exploded=exploded);//mecanum_wheel_rear_right(exploded=exploded);

/*check*/
// part_3_1: link_negative_x
translate([part_3_1_x, part_3_1_y, part_3_1_z])
link_negative_x(exploded=exploded);

/*check*/
// part_3_2: link_negative_z
translate([part_3_2_x, part_3_2_y, part_3_2_z])
link_negative_z(exploded=exploded);

/*check*/
// part_3_3: joint_negative_x
translate([part_3_3_x, part_3_3_y, part_3_3_z])
joint_negative_x(exploded=exploded);

/*check*/
// part_3_4: wheel_left
translate([part_3_4_x, part_3_4_y, part_3_4_z])
wheel_left(exploded=exploded);//mecanum_wheel_rear_left(exploded=exploded);

/*check*/
//part_4:chassis
translate([part_4_x,part_4_y,part_4_z])
chassis(exploded=exploded);

/*check*/
//part_5:chassis
translate([part_5_x,part_5_y,part_5_z])
chassis(exploded=exploded);

/*check*/
// part_6_1: link_positive_x
translate([part_6_1_x,part_6_1_y,part_6_1_z])
link_positive_x(exploded=exploded);

/*check*/
// part_6_2: joint_negative_z
translate([part_6_2_x,part_6_2_y,part_6_2_z])
joint_negative_z(exploded=exploded);

/*check*/
// part_6_3: joint_positive_x
translate([part_6_3_x,part_6_3_y,part_6_3_z])
joint_positive_x(exploded=exploded);

/*check*/
// part_6_4: wheel_right
translate([part_6_4_x,part_6_4_y,part_6_4_z])
wheel_right(exploded=exploded);//mecanum_wheel_front_right(exploded=exploded);

/*check*/
// part_7_1: link_negative_x
translate([part_7_1_x,part_7_1_y,part_7_1_z])
link_negative_x(exploded=exploded);
```

```
/*check*/
// part_7_2: joint_negative_z
translate([part_7_2_x,part_7_2_y,part_7_2_z])
joint_negative_z(exploded=exploded);

/*check*/
// part_7_3: joint_negative_x
translate([part_7_3_x,part_7_3_y,part_7_3_z])
joint_negative_x(exploded=exploded);

/*check*/
// part_7_4: wheel_left
translate([part_7_4_x,part_7_4_y,part_7_4_z])
wheel_left(exploded=exploded);//mecanum_wheel_front_left(exploded=exploded);
```

Now I will start the input, please complete your task according to the above requirements.