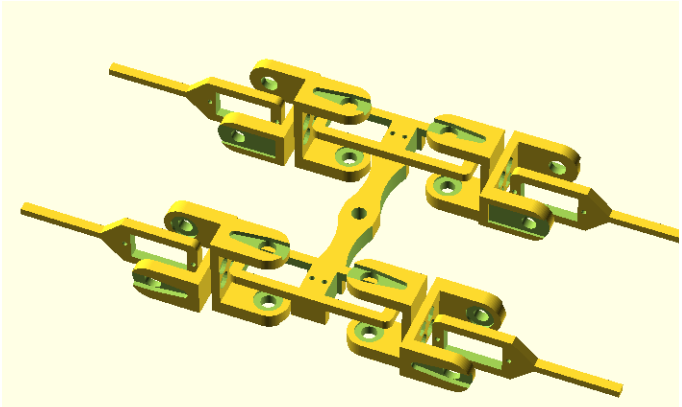
1. le squelette – impression 3d

Les pièces du robot sont dessinées sur ordinateur puis fabriquées sur l'imprimante 3d.
Nous avons récupéré une base de squelette de quadrupède sur le site thingiverse.com/thing:38159



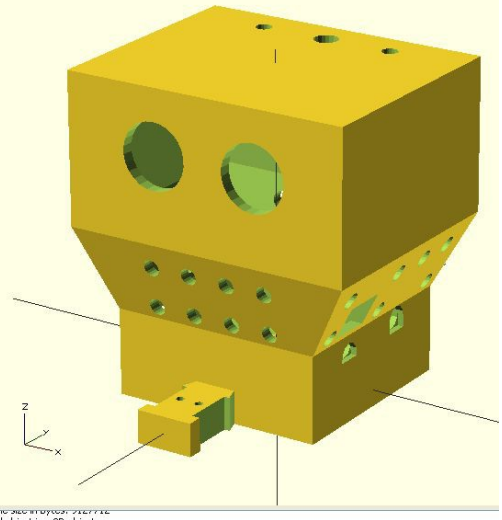
Que nous avons adapté pour prendre en compte les dimensions de nos servomoteurs.

Puis nous avons créé, avec le logiciel OpenSCAD, une base et une tête, sur mesure pour accueillir la carte électronique, les fils, capteurs et piles.

OpenSCAD - centre-boite-2015-04-30.scad*

File Edit Design View Help

```
module corps() {

   // partie assemblage avec chappe
   difference(){
      translate([-26,-25,11]) color("purple") cube([52,50,6]);
      translate([-23,-22,11]) cube([46,44,6]);
      translate([-24.5,-23.5,11]) cube([49,47,3]);
      translate([-50,-10,12]) rotate([0,90,0]) cylinder(d=6,h=100);
      translate([-50,10,14]) rotate([0,90,0]) cylinder(d=6,h=100);
      translate([-50,7,11]) cube([90,6,3]);
      translate([-50,-13,11]) cube([90,6,1]);
   }

   // gros cube
   difference(){
      translate([-36,-29,36]) cube([72,58,35]);
      translate([-33,-26,36]) cube([66,52,35]);
      translate([-13,-29.5,57]) rotate([0,90,90]) cylinder (d=16.2,h=4);
      translate([13,-29.5,57]) rotate([0,90,90]) cylinder (d=16.2,h=4);
   }

   //encoches batterie

   translate([-33,16.5,36]) cube([2,1,15]);
   translate([-33,-17.5,36]) cube([2,1,15]);

   translate([23,16.5,36]) cube([10,1,15]);
   polyhedron(
         points=[
               [23,16.5,51],[33,16.5,51],[33,16.5,65],
               [23,17.5,51],[33,17.5,51],[33,17.5,65]
         ],
         faces=[
               [0,1,2],[4,3,5],[3,0,2,5],[4,1,0,3],[1,4,5,2]
         ]
      );

   translate([23,-17.5,36]) cube([10,1,15]);
   polyhedron(
         points=[
               [23,-17.5,51],[33,-17.5,51],[33,-17.5,65],
               [23,-16.5,51],[33,-16.5,51],[33,-16.5,65]
         ],
         faces=[
```

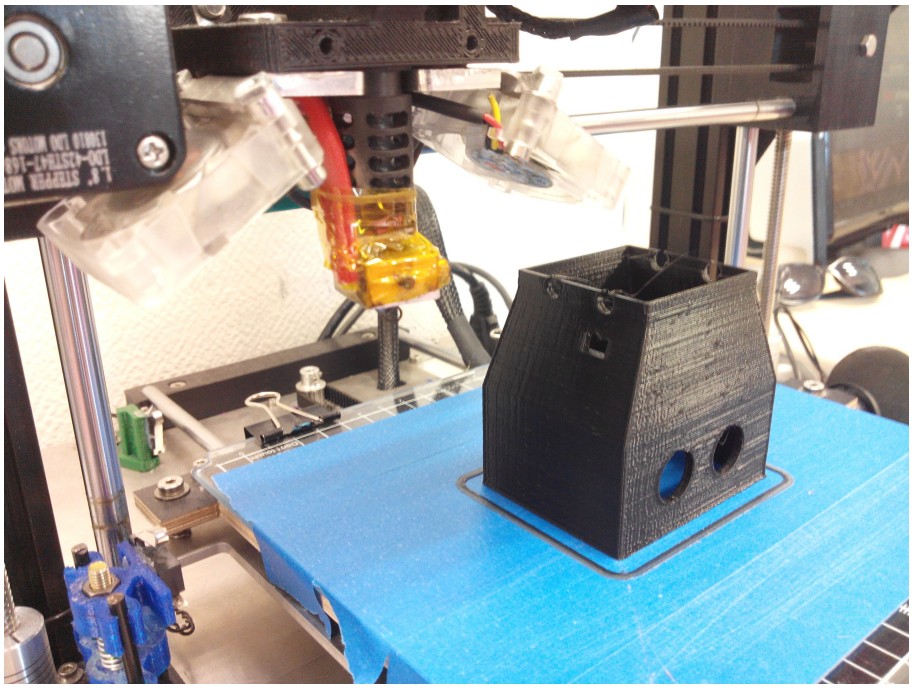


Top level object is a 3D object:
Simple: no
Vertices: 2032
Halfedges: 6064
Edges: 3032
Halffacets: 1994
Facets: 997
Volumes: 2
Total rendering time: 0 hours, 0 minutes, 34 seconds
Rendering finished.

Viewport: translate = [2.58 13.17 36.49], rotate = [69.70 0.00 32.70], distance = 762.08

Après avoir dessiné les éléments, il nous reste à les fabriquer grâce à l'imprimante 3d.

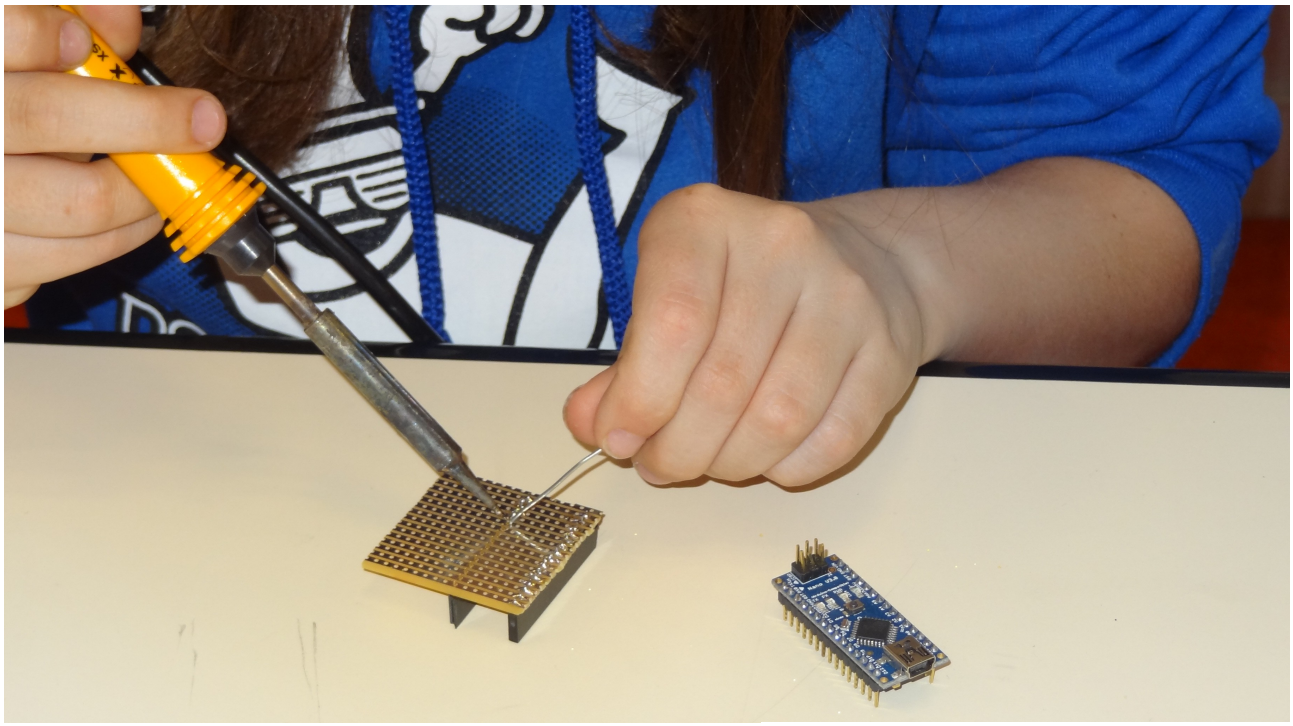
Les pièces ne sont pas homogènes car nous n'avons plus le même fil plastique que celui avec lequel nous avions commencé.



2. l'électronique, les fils, la soudure

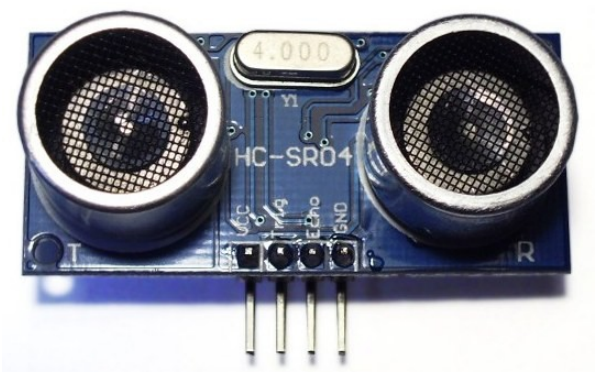
La logique du robot est intégrée dans une carte arduino. Nous avons choisi la version arduino nano, car c'est une toute petite carte, plus facile à intégrer dans un petit objet robotique.

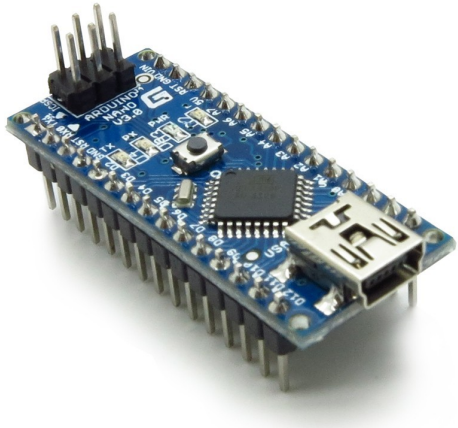
Nous avons utilisé une carte de connexion pour souder les fils des servomoteurs et de la batterie à la carte arduino.



Composants :

8 servomoteurs
1 sonar HC-SR04
1 carte arduino nano
4 piles AA 1,2V 2700ma
1 interrupteur





3. le programme arduino

Les cartes open source arduino se programment grâce à une application elle aussi open source : il s'agit du programme **arduino**, et selon wikipedia "le langage de programmation utilisé est le C++, compilé avec avr-g++ 3, et lié à la bibliothèque de développement Arduino, permettant l'utilisation de la carte et de ses entrées/sorties."

Après de nombreux tests, nous avons mis au point **plusieurs fonctions** :

- calculDistance()

Cette fonction permet d'interroger le sonar pour connaître la distance, permettant ainsi de **détecter les obstacles, les murs**.

- avance()

La fonction avance() a été la plus compliquée à écrire : il faut donner des positions à chaque servomoteur, en les enchaînant correctement pour que les 4 pattes du quadripède effectuent un **mouvement de marche**. Nous avons regardé des vidéos d'animaux à quattre pattes, nous avons décomposé le mouvement à obtenir, puis codé la fonction. Après de nombreux tests et un résultat loin d'être satisfaisant, nous avons récupéré un code de marche de quadrupède sur **github**, que nous avons adapté pour correspondre à notre quadrupède, et là le résultat est satisfaisant.

- demiTour()

Le demi-tour est obtenu en reprenant la fonction avance(), et en inversant l'ordre des instructions concernant le côté gauche du robot. Ainsi les pattes de droite avancent, les pattes de gauche reculent, donc **le robot pivote**.

- debout()

Avant de se mettre à marcher, le robot doit se lever. Nous nous sommes amusés à lui faire **lever les pattes** avant, puis les pattes arrière.

- hautLesMains()

La fonction hautLesMains() est utilisée quand le robot détecte un **obstacle très proche**, et doit donc s'arrêter. Il lève les pattes en l'air, ce qui le place dans une position permettant d'actionner l'interrupteur, et de ranger facilement le robot.

Le **comportement du robot** utilise ces différentes fonctions :

1 caculDistance()
2 si la distance est inférieure à 20cm, hautLesMains()
3 sinon, si la distance est inférieure à 40cm, demiTour()
4 sinon, debout() et avance()