Session 1

1. Chassis and hull design.

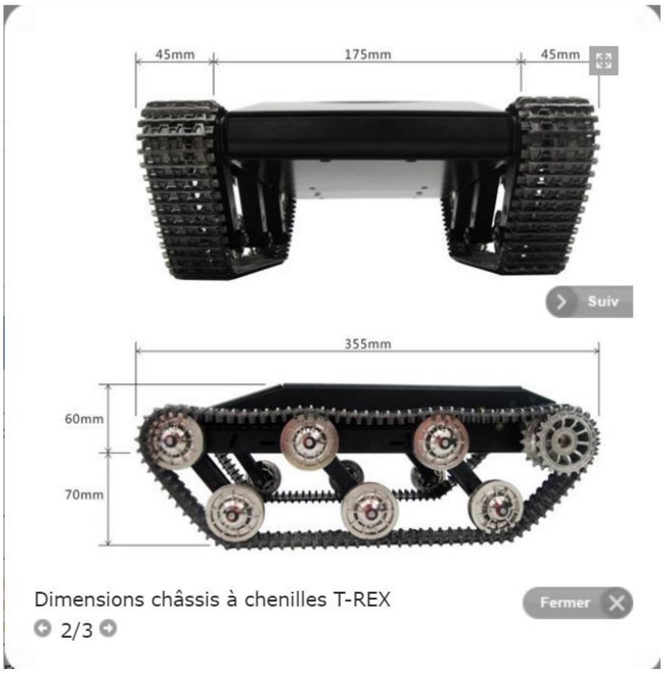
The first part of the session was dedicated to the search for the crawlers that best suited the structure of the imagined robot, after many counting, we decided to start on the model T-rex 355*265*130 mm, which allowed to obtain a certain exploitable volume.

The track system will be recovered and we will inject our own DC motor.

https://www.gotronic.fr/art-chassis-a-chenilles-t-rex-21385.htm

Once the measurement were established, the first sketch of the hull was made.

It was necessary to determine the most optimal location of the components inside the hull (a wooden cube 175*350*70), the rest of the height is left to the retractable arm.



Dimensions châssis à chenilles T-REX
2/3

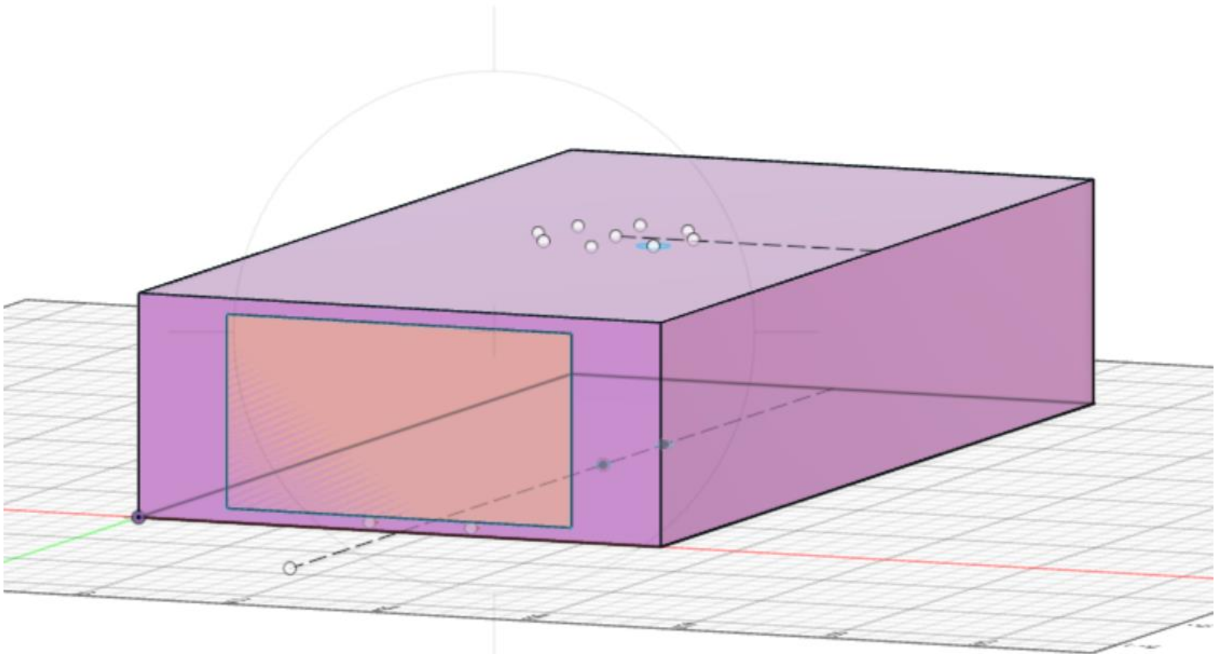
2. Arrangement of the components within the shell.

The 6 wooden boards will be connected by screwed gaps otherwise the ventilation nestled on the back will be neglected at first.

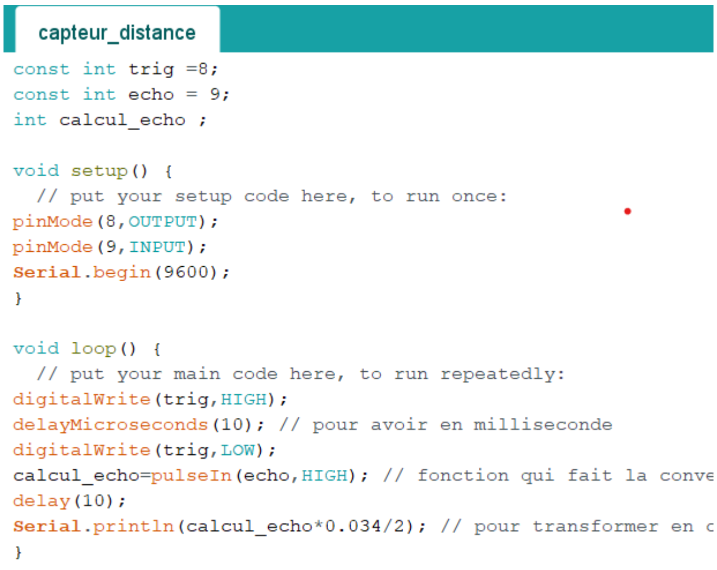
- Top board: the arm will be connected to the rest of the structure by a circular system of 8 holes.
- Back board: the battery whose mass estimated at 70g will be glued to the middle of the rear board.
- Front board: a 3mm plexiglass plate will be plated to the front board, through this plate the camera supported by a modeled support *to create the next session*, the ultrasound sensor and infrared detector will be able to work.

- Bottom board: will contain the rest of the electronic components (accelerometer/gyroscope/Arduino board screwed in the middle, Nvidia card connected initially to the camera) by adding holes to connect all the actuators and electronics, which will be glued with hot glue or tape).

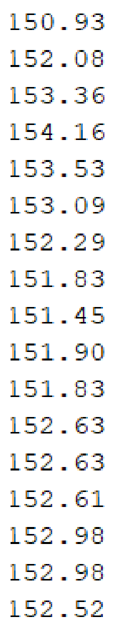
Once all the components delivered, we will make sure of the measurements again before cutting the frame in the final material (wood).



Once the chassis / shell part is finished, I've been interested in operating the HC-SR04 ultrasound sensor that will be glued to the bottom board back of the plexiglass plate.

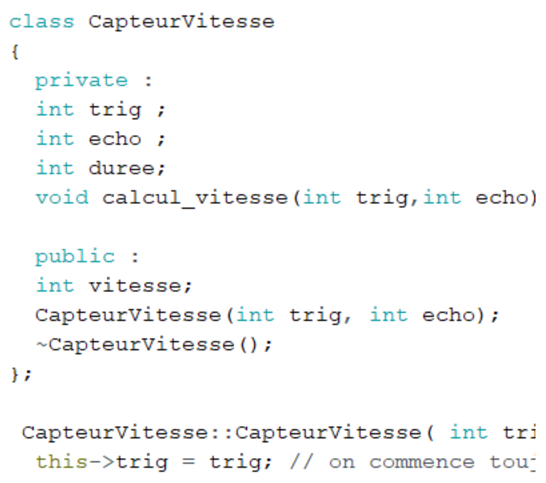


```
const int trig =8;
const int echo = 9;
int calcul_echo ;

void setup() {
  // put your setup code here, to run once:
pinMode(8,OUTPUT);
pinMode(9,INPUT);
Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(trig,HIGH);
delayMicroseconds(10); // pour avoir en milliseconde
digitalWrite(trig,LOW);
calcul_echo=pulseIn(echo,HIGH); // fonction qui fait la conve
delay(10);
Serial.println(calcul_echo*0.034/2); // pour transformer en c
}
```



```
150.93
152.08
153.36
154.16
153.53
153.09
152.29
151.83
151.45
151.90
151.83
152.63
152.63
152.61
152.98
152.98
152.52
```

Well, the sensor works, I can now define a SensorDistance class that can be called anytime during the program to display the speed.



```
class CapteurVitesse
{
  private :
  int trig ;
  int echo ;
  int duree;
  void calcul_vitesse(int trig,int echo)

  public :
  int vitesse;
  CapteurVitesse(int trig, int echo);
  ~CapteurVitesse();
};

 CapteurVitesse::CapteurVitesse( int tri
  this->trig = trig; // on commence tou
```