

**DEUXIÈME ANNÉE DE BUT
GÉNIE ÉLECTRIQUE ET
INFORMATIQUE INDUSTRIELLE**

IUT D'ÉVRY VAL D'ESSONNE

**CONCOURS ROBOTIQUE
DES IUT DE GEII**

CACHAN

Année 2023 - 2024

Préparé par :

Aubin TOURAIS

&

Matthieu MOCHET

Enseignant :

Hicham HADJ-ABDELKADER

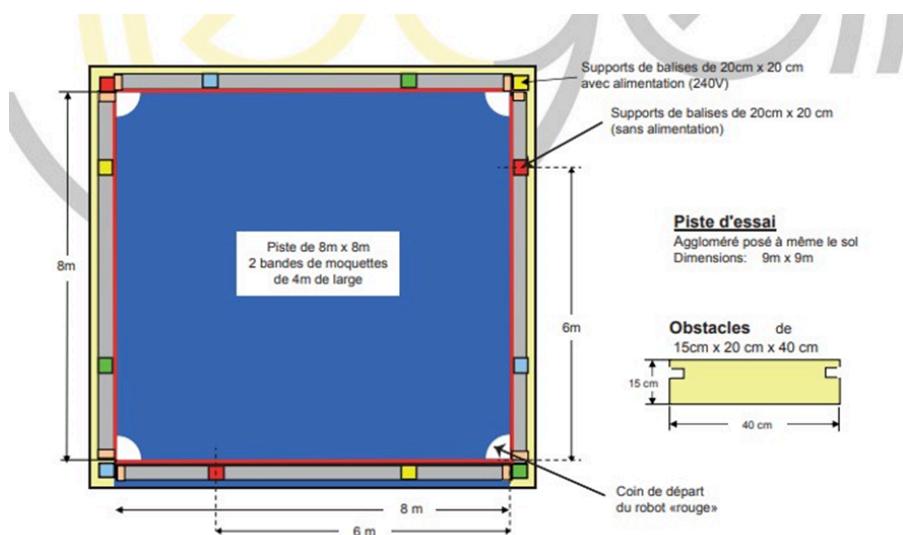
SOMMAIRE

Cahier des Charges.....	3
● Règles :	4
Stratégies.....	6
I. Schéma fonctionnel.....	6
II. Les différentes tâches :	6
● Tâche 1 : Calcul et choix du moteur.....	6
● Tâche 2 : Stratégie mise en place pour le déplacement du robot.....	7
● Tâche 3 : Listes des composants nécessaire pour le projet.....	8
● Tâche 4 : Schématisation de la coque du robot.....	9
● Tâche 5 : Conception de la coque du robot.....	11
● Tâche 6 : Conception des balises de référence.....	11
● Tâche 7 : Programmation du robot.....	13
Programmation Carte Arduino.....	20
● Fonctionnement des moteurs :	20
● Utilisation des Télémètres :	21
● Le Capteur de couleur :	23
● L'émission Infrarouge (Balise) :	24
● La réception Infrarouge :	24
● Le servomoteur (Éclatement Ballon) :	26
● Le démarrage Jack :	27
● Le programme final (union de toutes les fonctions) :	27
Jours du concours.....	35
● Jeudi Après-midi :	35
● Vendredi Matin :	35
● Vendredi Après-midi :	36
● Samedi Matin :	37
● Samedi Après-midi :	37
Résultat.....	38
Retour sur Expérience.....	39
● Matthieu MOCHET :	39
● Aubin TOURAISS :	39
Conseils et remarques.....	41
Annexes.....	43

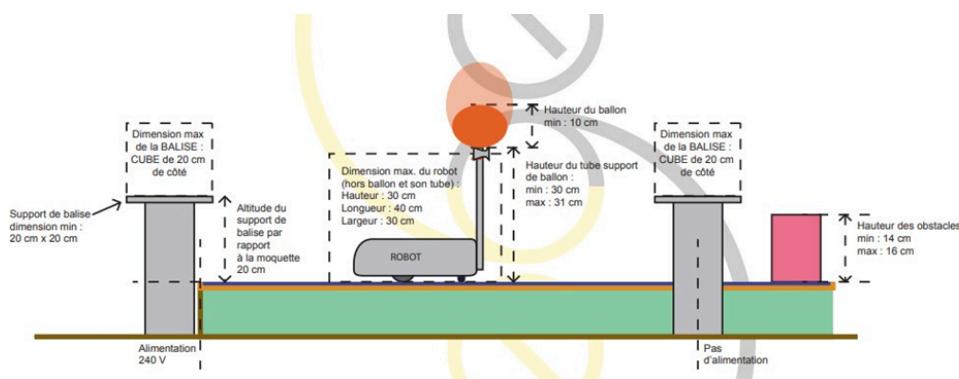
Cahier des Charges

Le but est de réaliser un robot autonome pouvant aller d'un point A à un point B, se trouvant à sa diagonale, sur un terrain de 8 mètres par 8 mètres avec des obstacles (faisant du 15cm x 20cm x 40cm) se trouvant à différents endroits du terrain. Nous avons de plus la possibilité de pouvoir placer des balises dans 3 coins du terrain préinstallé afin de pouvoir aider le robot à se repérer, avec une se trouvant au niveau de la zone d'arrivée du robot.

Voici un exemple plus concret du terrain où se trouvera notre robot.



Dimensions du parcours et placement des balises. Source : Sujet concours Cachan



Dimensionnement des éléments du parcours et du robot. Source : Sujet concours Cachan

- **Règles :**

Pour pouvoir faire la conception du robot, nous avions la restriction de devoir faire un robot sous les dimensions suivantes :

- Longueur du robot : 40cm
- Largeur du robot : 30cm
- Hauteur du robot : 30cm

De plus, nous devions mettre en place un éclatement de ballon lorsque le robot atteint la zone de fin (le point B). Notre robot devra aussi passer une phase d'homologation, pour qu'un robot soit homologué, il doit effectuer un parcours avec un départ au jack et un arrêt dans la zone d'arrivée, seul sur la piste normale, sans obstacle au milieu en moins de 90 secondes.

Il y a de plus un système de comptage de points permettant ainsi de savoir si le robot répond bien aux attentes et qu'il effectue bel et bien sa mission. On se reportera, pour plus de détails, à l'**annexe A.1 (p. 43)**.

Afin de pouvoir faire cela, nous devions faire la schématisation de l'intégralité de notre robot avant même de nous lancer dans ce projet.

Nous nous sommes donc concentrés sur un maximum de points, c'est pourquoi nous sommes partis sur l'idée de faire un lapin comme coque de notre robot pour représenter la fluidité du lapin et sa rapidité (voir annexe). Mais pour cela, nous devions tous prendre en compte, la dimension des roues du robot, car nous devions choisir nous-même les roues, les moteurs du robot, la conception du châssis du robot, sa coque, et la conception des balises permettant de guider le robot.

C'est à partir de là que nous avons coupé l'intégralité de la conception sous plusieurs tâches :

- Calcul ainsi que le choix du moteur le plus adapté
- Stratégie mise en place pour le déplacement du robot
- Listes des composants nécessaires pour le projet
- Schématisation de la coque du robot
- Conception de la coque du robot
- Conception des balises de référence
- Programmation du robot

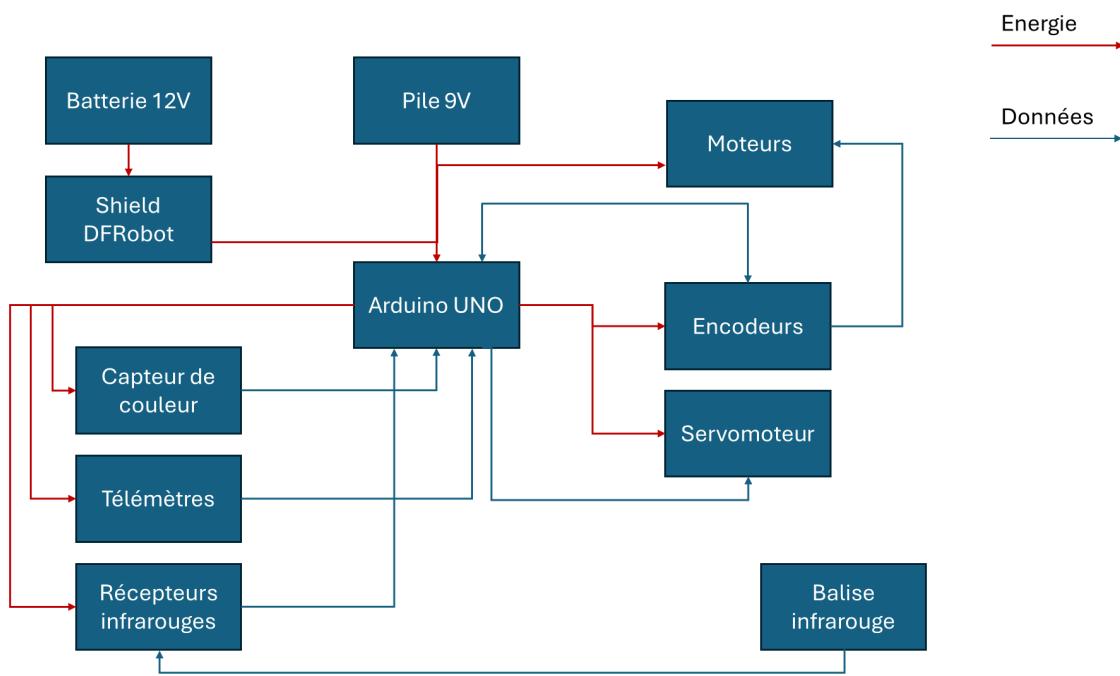
Chaque point est important afin de pouvoir bien se lancer dans la conception du robot, afin de ne pas perdre de temps, et d'avoir ainsi le robot le plus performant possible.

Il est important de noter, qu'en attendant de faire la conception de notre châssis, de recevoir nos roues pour notre robot qui prend beaucoup de temps, nous avions à disposition le robot du concours de l'an dernier des étudiants de BUT +3, afin de pouvoir y faire nos tests sur la programmation et de ne pas perdre de temps. Malgré cela, nous avons refait le robot de zéro, à notre manière et sans s'appuyer sur celui-ci.

C'est ainsi que nous allons voir en détail chaque tâche, nous permettant de voir l'avancée du robot et de constater comment nous nous y sommes pris.

Stratégies

I. Schéma fonctionnel



II. Les différentes tâches :

- **Tâche 1 : Calcul et choix du moteur**

Comme première tâche, nous devions faire un calcul afin de pouvoir définir les roues qui nous seront les plus utiles afin de pouvoir réaliser le parcours le plus aisément possible. Il faut savoir que de base, nous utilisons les moteurs et les roues que le groupe de l'an dernier avait utilisé pour leur concours, néanmoins, ces roues ne tiennent vraiment pas le rythme, patine sur le sol et quand nous avons testé sur la moquette que nous avions à disposition, qui est similaire à celui du concours, le robot avait beaucoup de mal à avancer.

Nous avons donc pris en compte ce facteur, et avons décidé de ne pas prendre ces roues, de changer aussi les moteurs afin d'avoir plus de couple permettant à notre robot de pouvoir se déplacer plus facilement et plus rapidement.

De plus, nos anciens camarades, nous ont signalé que la moquette présente au sein du concours était plus agrippante et que leur robot avançait mieux, malgré cela, nous voulions un robot qui était sur de pouvoir tenir, et voulions être sûr que celui-ci ferait l'affaire, si arrivé au jour du concours, notre robot va trop vite, nous pourrons modifier sa puissance directement via le code Arduino, cela n'est pas un problème. Vaut mieux qu'il soit plus puissant que l'inverse, cela serait problématique.

Pour cela, nous avons fait de nombreux essais sur divers moteurs, mais nous sommes arrivés à une conclusion d'un moteur plutôt intéressant qui est le suivant : Ensemble kit roue + motoréducteur + support 19E837RK. Nous sommes arrivés à ce résultat à la suite d'une suggestion de la part d'un superviseur M. CHEVREAU, qui a réalisé les calculs nécessaires en comparant les moteurs ainsi que les roues de l'an passé.

- **Tâche 2 : Stratégie mise en place pour le déplacement du robot**

Nous devons réfléchir à comment faire pour permettre au robot d'aller de son point de départ à son point d'arrivée en étant entièrement autonome.

Afin de pouvoir permettre au robot de pouvoir se déplacer en autonomie, nous utilisons une batterie de 12V pour l'alimentation des moteurs, permettant ainsi le déplacement du robot, et d'une pile de 9V, alimentant ainsi la carte Arduino afin de pouvoir effectuer un code qui sera chargé en elle au préalable.

Nous avons pour stratégie de faire repérer le robot via la balise se trouvant sur la piste à l'aide d'émetteurs et de récepteurs infrarouges. Le robot devra par la suite éviter les obstacles, pour cela, nous utiliserons des télémètres à ultrasons afin de pouvoir changer la direction de notre robot lorsqu'il détecte un obstacle à une certaine distance.

Nous utiliserons aussi un capteur de couleur, afin de pouvoir différencier le parcours d'obstacles avec la ligne de départ et la ligne d'arrivée, étant donné que notre robot sera à l'arrêt au départ et démarrera lorsque l'on lui retirera un câble jack qui est connecté à la carte Arduino, puis lorsque celui-ci arrivera sur la zone blanche de fin, il s'arrêtera.

Une fois arrêté, il devra éclater le ballon se trouvant sur lui à l'aide d'un servomoteur.

- **Tâche 3 : Listes des composants nécessaire pour le projet**

Afin de pouvoir réaliser ce projet, nous devions faire une liste des composants nécessaires pour la conception de notre robot.

- Capteur de couleur x1 : **TCS34725 [annexe B.1 (p.44)]**
- Télémètre à ultrason x3 : **HC-SR04 [annexe B.2 (p.44)]**
- Moteurs + roues x2 : **kit roue+motoréducteur+support 19E837RK [annexe B.3 (p.45)]**
- Émetteurs Infra-rouge x3 : **LED émettrice infrarouge 5 mm 940 nm [annexe B.4 (p.45)]**
- Récepteurs Infra-rouge x3 : **Récepteur infrarouge OS-1638 184291 [annexe B.5 (p.46)]**
- Servomoteur x1 : **servo S-0009 MG [annexe B.6 (p.46)]**
- Carte Arduino x1 : **Arduino Uno A000066 [annexe B.7 (p.47)]**
- Shield Carte Arduino x1 : **L298p shield v1.3 [annexe B.8 (p.47)]**
- Batterie de 12V x1 : **NP1.2-12 [annexe B.9 (p.48)]**
- Pile de 9V x1 : **Conrad energy Extreme Power Pile 6LR61 (9V) lithium 1200 mAh [annexe B.10 (p.48)]**
- Câble Jack x1 : **Clip de batterie 9V avec prise jack 5,5 mm [annexe B.11 (p.49)]**

Nous avons eu la majorité des composants qui nous ont été fournis par M. HADJ-ABDELKADER.

Celui-ci nous avait de même proposé d'autres composants, comme un gyroscope (Gyro Click L3GD20) permettant de mesurer la vitesse angulaire sur 3 axes, mais après avoir essayé de configurer celui-ci et avoir fait de multiples tentatives, nous n'arrivions pas à l'exploiter afin qu'il entre dans notre objectif de guider le robot sur la piste.

Afin de remédier à cela, nous sommes donc partis dans l'idée de guider notre robot uniquement avec des émetteurs et des récepteurs infrarouge en lien avec la balise. On se reportera, pour plus de détails, à l'**annexe C.7 à D.2 (p. 52 à 56)**.

- **Tâche 4 : Schématisation de la coque du robot**

Il s'agit avant tout de la partie sur laquelle nous n'aurions du prendre moins de temps avec la conception du robot, nous empêchant parfois de nombreux test et un résultat qui a été contre nous. Une chose à retenir ? **FAIRE SIMPLE et avoir un robot qui marche bien avant tout !**

Pour la schématisation de la coque du robot, il était primordial de savoir quelles roues nous devions utiliser, afin de prendre en compte leurs dimensions et les intégrer à nos calculs pour ne pas dépasser les limites imposées. Nous devions donc prendre les dimensions du robot qui nous était restreint.

Nous sommes partis dans l'idée, que pour être sûr de ne pas dépasser les dimensions du robot, nous avons conçu pour chaque cas, une diminution de 5cm, pour qu'ainsi nous soyons certains d'avoir un robot répondant aux attentes.

Il fallait dans un premier temps faire la conception de la base de notre robot, donc nous avions fait le châssis de notre robot, c'est sur cette partie que nous poserons notre carte Arduino UNO, notre batterie, notre pile, et que l'on viendra fixer nos moteurs et nos roues.

Par la suite, nous souhaitions que la coque de notre robot dépasse légèrement de notre châssis, permettant ainsi d'avoir un effet de relief et de pouvoir placer nos roues à la limite de la coque supérieure du robot.

Pour la partie supérieure de la coque du robot, nous sommes partis avec l'idée de faire un dos tout simplement arrondis qui recouvre ainsi les composants se trouvant dans notre robot.

Il y a une queue représentant ainsi le lapin, mais nous avons aussi rajouté un trou justement dans la queue, car il s'agit d'ici que nous tirerons le fil jack permettant le lancement du robot étant donné qu'il sera face à la piste.

De plus, sous la queue, nous avons ajouté un nouveau trou qui permet de pouvoir installer un bouton d'arrêt d'urgence, qui permettra d'arrêter notre robot si un problème survient, et cela nous était imposé par le cahier des charges.

Par le dessus de la coque nous retrouvons ensuite, une structure qui permet de pouvoir fixer notre ballon que nous devrons éclater à l'arrivée de notre parcours. Celle-ci composée d'un trou permettant ainsi de pouvoir fixer notre servomoteur qui effectuera l'éclatement du ballon lorsque l'Arduino lui donnera l'ordre.

La tête se trouve à l'avant, qui est fixé avec le haut du corps. C'est à l'intérieur que nous retrouverons les récepteurs infrarouges afin de pouvoir détecter la balise se trouvant sur la piste.

Sur le haut de la tête du lapin, nous avons ajouté un bloc arrondis permettant d'isoler les détecteurs infrarouges les uns des autres, afin qu'il puisse savoir où est-ce qu'il est par rapport à la balise pour qu'il puisse ensuite se rediriger vers cette dernière.

Nous avons, de plus, rajouté des pattes sur les côtés de notre lapin afin d'avoir une coque ressemblant un maximum à un vrai lapin, celles-ci, à l'arrière des roues. On se reportera, pour voir l'intégralité de la schématisation à l'**annexe C.6 (p. 52)**.

- **Tâche 5 : Conception de la coque du robot**

Pour la conception du robot, nous avons utilisé l'application AUTODESK FUSION 360 afin de représenter les parties de notre coque de robot pour ensuite la réaliser via une imprimante 3D appartenant à l'IUT.

La partie qui a été la plus longue et la plus difficile était de devoir apprendre à manipuler le logiciel, car n'ayant jamais utilisé d'application de modélisation de ce type, il fallait partir de zéro et tout apprendre. Il était important de prendre en compte chaque élément externe de la coque pour la conception afin de faciliter les branchements et les fixations.

Par exemple : Des trous sur le châssis, permettant ainsi la connexion des moteurs avec la carte Arduino Uno, ne se trouvant pas sur le même niveau, ou encore dans la tête du lapin, faire des trous permettant ainsi la mise en place des récepteurs infrarouge à la même hauteur que les émetteurs des balises.

Nous avions pris la décision de séparer la coque en plusieurs parties, afin de pouvoir faire l'impression de chaque pièce, les unes après les autres, pour être sûr que l'imprimante n'oublie pas un morceau de la coque et que le robot puisse être démontable.

Nous avons aussi fait en sorte d'ajouter des fixations pour pouvoir faire l'assemblage des pièces entre elles. Vous pourrez obtenir le résultat que nous avons fait pour la coque du lapin aux **annexes C1-C13 (p. 49 à 55)**.

- **Tâche 6 : Conception des balises de référence**

Pour la conception des balises de référence, nous sommes partis dans l'idée de faire des circuits imprimés pour nos capteurs faits main.

Nous avons donc réalisé le circuit électrique sur papier afin de pouvoir avoir un aperçu de ce que l'on souhaite obtenir, puis nous devions le remettre sur l'application WINSCHM, pour ensuite le transférer sur le logiciel WINTYPON en

respectant les contraintes de dimensions des différents composants. Mais avant de pouvoir se décider sur la conception de notre plaque pour les récepteurs, nous devions savoir où nous allions la placer sur notre robot, car nous avons certains points à prendre en temps avant cela.

La balise devra se trouver sur une plateforme se trouvant à 20cm du sol, nous devions donc avoir nos récepteurs à la même hauteur afin de pouvoir bien détecter nos balises lorsque notre robot se déplacera.

Il fallait donc prendre en compte le fait que nos récepteurs devront eux aussi être à 20cm en partant du sol, c'est ainsi que nous avons fait le choix de les positionner au niveau de la tête de notre robot, pour pouvoir faire la détection de nos balises par l'avant du robot et pouvoir par la suite, mieux se déplacer, et gagner quelques centimètres de distances.

Maintenant, il fallait se demander comment construire notre carte typon, nous avions pensé à les faire appart chacun et ainsi les fixer à notre guise, ou plutôt, faire une seule plaque pour tous nos récepteurs afin de pouvoir mieux les placer, et surtout les mettre à la même hauteur sans difficulté, nous pensions donc, à mettre cette plaque dans le crâne de notre lapin, afin de pouvoir détecter plus facilement sur une plus grande marge.

Mais un autre problème survenait, nos capteurs détectaient sur une trop grande marge en largeur, hors, nous voulions que chaque récepteur détecte en face de lui afin de mieux pouvoir déplacer notre robot et plus précisément. Pour résoudre cela, nous avons mis dans la conception 3D un cache avec des trous afin d'isoler les différents récepteurs, et ainsi pouvoir détecter vraiment la balise à chaque fois, en face du récepteur en question. Nous avions rencontré des problèmes à certains temps, où l'on pouvait détecter sans arrêt nos émissions via les récepteurs même quand nous voyions à l'opposé, cela était dû au mur se trouvant à l'arrière de notre robot quand nous faisions les tests, car cela reflétait les émissions d'infrarouge.

La balise aura pour mission de guider le robot afin de savoir où celui-ci se trouve sur le terrain, donc nous mettons en place sur la balise émettrice infrarouge. Cette balise sera alimentée via une carte Arduino afin de pouvoir recevoir le code qui lui correspond. Nous avons d'abord fait la conception de l'émetteur sur une plaque à trou pour nos tests puis réalisé un typon et une impression 3D du boîtier, pour plus de détails, aux **annexes C.12 C.13 et D.1, (p. 55 et 56)**. Et pour les récepteurs, il s'agit du même cas, pour plus de détails, aux **annexes C.7, C.8, C.9, C.10 et D.2 (p. 52, 53, 54 et 56)**.

● **Tâche 7 : Programmation du robot**

La programmation du robot doit être faite en langage Arduino. Nous avions à disposition une carte Arduino UNO 3, ainsi qu'une autre partie se trouvant sur la carte qui est un SHIELD, cela permettra de pouvoir faire directement la liaison des moteurs de notre robot sur la carte.

Pour la partie programmation, nous avons dans un premier temps fait la programmation de chaque composant les uns après les autres, car nous devions savoir quelle bibliothèque était nécessaire pour chaque composant, les tester un par un, afin d'être sûr que les composants effectuent les tâches que nous avions décidé de leur attribuer.

Voici comment nous avons pensé pour l'utilisation de nos composants sur nos robots :

- Moteurs : faire avancer/tourner le robot;
- Servomoteur : faire tourner son axe pour faire éclater le ballon;
- Capteur de couleur : savoir si nous sommes sur la zone de départ/arrivée, ou sur le circuit de parcours;
- Emetteur infrarouge : envoyer des données via communications sans fil aux récepteurs afin de diriger le robot;
- Récepteur infrarouge : recevoir des communications sans fil depuis les émetteurs afin de diriger le robot;
- Prise Jack : permettre le lancement du robot au départ;
- Télémètres à ultrasons : savoir s'il y a un obstacle sur le chemin.

Après les avoir configurés les uns après les autres, nous devions effectuer divers tests afin d'assembler chaque composant ensemble, permettant ainsi de voir la compatibilité entre chacuns.

Par exemple : le moteur avec les télémètres, pour changer la direction du robot si nous détectons un obstacle sur le chemin.

Une fois la liaison de chaque composant entre eux, nous devions faire l'assemblage intégral de chaque composant entre eux, afin d'avoir un seul code pour le robot.

Il faut noter, que nous avons fait un pseudo-code au préalable, afin de savoir comment nous y prendre, et savoir ainsi quel composant est prioritaire sur quel composant.

Nous sommes arrivés à la conclusion suivante pour l'ordre d'exécution :

- La prise Jack qui permet le lancement du robot;
- Démarrage des roues durant 0.5s afin de quitter la zone blanche de départ;
- Détection de la piste bleue, donc nous pouvons lancer le robot jusqu'à ce qu'il détecte une zone blanche qui doit être la zone finale;
- Détection d'un obstacle via un des télémètres, nous devons donc changer l'orientation du robot pour l'éviter;
- Détection de la balise, nous essayons de nous diriger toujours vers la balise finale;
- Détection de la piste blanche, nous arrêtons donc le robot, et nous faisons l'éclatement du ballon via le servomoteur, avec une série de 3 exécutions afin d'être sûr d'éclater le ballon;
- Nous arrêtons l'intégralité du robot.

Le robot devra être capable d'éviter des obstacles, et pour arriver à cela, nous utiliserons les télémètres à ultrasons, qui évalueront la distance qu'il y a entre l'obstacle en question et le robot.

Voici les choix que nous avons fait afin de pouvoir éviter l'obstacle selon le capteur qui le détecte :

- Télémètre Gauche détecte : nous tournons vers la droite
- Télémètre Avant détecte : décalage vers la gauche
- Télémètre Droite détecte : nous tournons vers la gauche

Nous avons aussi pour chaque balise, un code se trouvant dans une carte Arduino, envoyant des données à notre robot qu'il peut lire via les récepteurs infrarouges.

Chaque balise envoie une donnée qui lui est configurée afin de pouvoir permettre au robot de savoir quelle balise lui envoie quelle information, cela nous permet de savoir où se trouve le robot sur la piste, car nous avons fait en sorte d'avoir toujours la priorité sur la balise finale.

Afin d'avoir plus de chance de pouvoir recevoir la détection d'une balise mais surtout de la balise finale, nous avons mis en place plusieurs récepteurs infrarouge sur la tête du lapin afin d'être persuadé de pouvoir toujours en détecter une :

- Récepteur Gauche : Tourner vers la gauche;
- Récepteur Avant : Toujours tout droit;
- Récepteur Droite : Tourner vers la droite.

Nous sommes arrivés à cette décision, car nous savons que les roues des moteurs ne demandent pas la même puissance, donc cela peut arriver qu'une roue soit plus puissante que l'autre, nous préférons donc avoir la certitude de toujours pouvoir détecter une balise.

Si dans le pire des cas, nous ne détectons aucune balise, car elles se trouvent à dos au robot, nous avons fait en sorte de faire tourner le robot sur lui-même afin de pouvoir avoir l'opportunité de détecter à nouveau une balise, et ainsi, reprendre sa course.

Pour la mise en place des récepteurs infrarouges, nous avons rencontré un problème qui nous a fait obstacle durant notre conception du code final. En effet, lorsque nous avons fait l'attribution des entrées à nos composants, nous avions fait le test individuellement de nos trois récepteurs et nous avons pu voir qu'ils marchaient tous.

Sauf que, lorsque nous les avions rassemblés dans le même code, nous avons pu constater que seulement un seul récepteur était opérationnel, ce qui n'était pas normal, étant donné que tous marchaient avec la même configuration .

Nous avons pu constater après avoir fait plusieurs tentatives, que lorsque nous configurons les trois récepteurs, uniquement le dernier récepteur est pris en compte, car, dans la bibliothèque que nous avions prise pour le programme des émetteurs ainsi que récepteurs infrarouges, nous ne pouvions faire la configuration que d'un récepteur sur un programme.

Afin de parvenir à résoudre ce problème, nous avons essayé de modifier le fichier source de la bibliothèque afin d'y ajouter plus de possibilités, de configurer nos récepteurs directement dans un fichier à part en les appelant les uns après les autres, mais cela ne changeait rien. Nous avons pu trouver une solution sur un forum Arduino, où une personne avait exactement le même problème que nous, il expliquait qu'il voulait en programmer un certain nombre et que cela se limitait à un seul récepteur.

Dans l'article sur le forum, la personne en question a réussi à obtenir la bibliothèque que nous utilisions avec une mise à jour plus récente, ce qui nous a résolu le problème des récepteurs. Après avoir changé les modifications dans la bibliothèque, nous avons refait le test sur notre code, et celui-ci était enfin en fonctionnement, nous pouvions donc enfin gérer nos cinq récepteurs sur notre carte Arduino en simultané.

Pour finir, nous avions un problème qui parvenait sans arrêt durant les essais sur notre code final, il s'agissait du temps de réponse et d'exécution de notre code, qui était lent, ce qui est dérangeant pour la piste, car si nous rencontrons un obstacle à une certaine distance, et que notre robot met beaucoup de temps avant de se décider à l'éviter, nous perdrons du temps et risquerons de rentrer dans l'obstacle en question.

Après de nombreux tests, nous avons pu constater qu'il s'agissait d'un problème venant de la bibliothèque directement, car elle lisait les informations à des délais différents ce qui bloquait nos réceptions. Nous avons donc utilisé une nouvelle bibliothèque qui prend en compte tous les télémètres d'un coup, et nous pouvons jouer sur la vitesse de réception avec des delay() ce qui est plus efficace que l'ancienne.

Il y avait un autre problème que nous avons rencontré, il s'agissait du moment où nous faisions la compilation de notre code afin de faire nos tests, nous avions parfois un message qui nous disait comme quoi la carte n'est pas reconnue par l'ordinateur et donc le chargement n'est pas possible. Nous avons pu constater qu'il s'agissait du moment où nous avons ajouté les cinq récepteurs, car nous avions attribué deux récepteurs sur les entrées numériques 0 et 1, qui sont de base réservé à la transmission de données ainsi qu'à la réception.

Nous devions donc lorsque nous faisions un chargement, débrancher ces récepteurs, attendre que le chargement du code soit terminé, puis les rebrancher, ce qui n'était pas très pratique.

Nous avons donc voulu changer de bibliothèque comme pour les télémètres, mais nous avions un message d'erreur lors de la compilation.

Ce qui signifiait que nous avions le même appel de fonctions dans deux bibliothèques différentes, ce qui est impossible à lire pour une compilation. Pour

résoudre cela, après avoir lu toutes les lignes de code des deux bibliothèques, nous avons modifié le fichier source de la bibliothèque que nous venions d'ajouter, au niveau des TIMER.

Dans la bibliothèque des récepteurs et des télémètres, nous avions des fonctions similaires qui étaient le TIMER2.

Nous avons donc modifié celui-ci dans la bibliothèque des récepteurs en TIMER5, étant donné qu'il n'était pas utilisé dans l'entièreté des bibliothèques :

```
316     #if defined(__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).  
317     ISR(TIMER4_OVF_vect) {  
318         intFunc(); // Call wrapped function.  
319     }  
320     #elif defined(__AVR_ATmega8__) || defined(__AVR_ATmega16__) || defined(__AVR_ATmega32__) || defined(__AVR_ATmega8535__)  
321     ISR(TIMER5_COMP_vect) {  
322         intFunc(); // Call wrapped function.  
323     }  
324     #elif defined(__arm__)  
325     // Do nothing...  
326     #else  
327     ISR(TIMER5_COMPA_vect) {  
328         intFunc(); // Call wrapped function.  
329     }  
330     #endif
```

Morceau de code dans la bibliothèque IRremote : Equipe IUT Evry

Au résultat, tout marchait correctement, et nous pouvions donc rajouter autant de récepteurs que nous souhaitions pour le jour du concours.

Puis vint le jour de la finale, nous avons fait de nombreux tests, recalibrage avec la luminosité etc. Nous avons déjà pu constater que la balise émettait comme il faut, aucune réflexion et notre code permettait à notre robot de ne jamais se mélanger avec un autre signal d'une autre balise. Un point réussi !

Malgré cela, notre code pour le capteur de couleur était trop complexe étant donné qu'il faisait une moyenne de la valeur du RGB pour ensuite en déduire la valeur du bleu. Après avoir vu avec notre professeur, un code RGB du TCS se trouvant en exemple de la bibliothèque était bien plus performant et surtout bien plus simple (une simple lecture).

Nous avions la même avec un problème de lecture des télémètres qui parfois lisait 0 alors qu'un obstacle se trouvait devant lui. C'est en prenant un programme simple dans les exemples, que nous avons pu corriger ce problème. Mais nous aurions dû faire plus attention au préalable, cela était notre faute.

Le programme fonctionne de la manière suivante, au départ le robot est sur la zone blanche et l'état du jack est à 1, une fois le jack débranché, son état passe à 0 et il avance jusqu'à la zone bleue.

Par la suite, le robot doit se débrouiller avec les télémètres et les récepteurs infrarouges. Les télémètres ont la priorité de détection car s'il y a un obstacle on ne veut pas qu'il se le prenne, ensuite, c'est le récepteur avant qui a la priorité, la balise finale attire le robot, puis ce sont les récepteurs gauches et droites.

Si le robot ne trouve pas d'obstacle et qu'il ne trouve pas de balise dans son champ de vision, il avance tout droit jusqu'à retrouver la balise afin qu'il exécute le code comme il faut.

Pour reconnaître la balise, cette dernière envoie une trame, nous avons choisi de lui faire envoyer Avant, cette trame nous l'avons traduite en ASCII puis passée en hexadécimale afin qu'elle puisse être utilisée par le programme Arduino.

Programmation Carte Arduino

• Fonctionnement des moteurs :

Pour ce programme, nous faisons tout simple la configuration des moteurs, qui sont reliés directement avec le Shield de notre carte, reliant les sorties PWM aux moteurs directement. Nous configurons donc les sorties, puis faisons une fonction permettant au robot d'avancer.

```
#include <avr/io.h>
#include <Wire.h>

int Moteur_Gauche = 5; // M1+
int Sens_Gauche = 4; // S1
int Moteur_Droite = 6; // M2+
int Sens_Droite = 7; // S2

// CONFIGURATION
void setup() {
    Serial.begin(9600);
// Place les pins des 2 moteurs et leurs sens en sortie
    pinMode(Moteur_Gauche, OUTPUT);
    pinMode(Sens_Gauche, OUTPUT);
    pinMode(Moteur_Droite, OUTPUT);
    pinMode(Sens_Droite, OUTPUT);
}

// PROGRAMME PRINCIPAL
void loop() {
    avancer(); // Appel de la fonction avancer pour bouger les moteurs
}

// FONCTIONS
void avancer(){
    digitalWrite(Sens_Gauche, 1); // Choix sens du moteurs vers l'avant
    digitalWrite(Sens_Droite, 1);

    analogWrite(Moteur_Gauche, 100); // Valeur allant de 0 a 255
    analogWrite(Moteur_Droite, 100);
    delay(2000);

    analogWrite(Moteur_Gauche, 0); // Arret des moteurs
    analogWrite(Moteur_Droite, 0);
    delay(2000);
}
```

- **Utilisation des Télémètres :**

Pour ce programme, nous faisons tout simple la configuration des télémètres. Nous configurons donc les sorties, puis faisons une fonction testant toutes les conditions depuis une table de vérité.

- **Table de Vérité : (Entrées = Télémètres / Sorties = Exécution Moteurs)**

Gauche	Avant	Droite	Tourner à Gauche	Avancer	Tourner à Droite	Reculer
0	0	0		1		
0	0	1	1			
0	1	0			1	
0	1	1	1			
1	0	0			1	
1	0	1		1		
1	1	0			1	
1	1	1				1

```
#include <avr/io.h>
#include "Ultrasonic.h"

Ultrasonic ultrasonic1(13);
Ultrasonic ultrasonic2(12);
Ultrasonic ultrasonic3(11);

#define Min_Obstacle 2
#define Detection_obstacleMax 17
#define Detection_obstacleMaxAvant 30

// CONFIGURATION
void setup() {
    Serial.begin(2400);
}

// PROGRAMME PRINCIPAL
void loop() {
    long TelGauche = ultrasonic1.MeasureInCentimeters();
```

```

long TelAvant = ultrasonic2.MeasureInCentimeters();
long TelDroite = ultrasonic3.MeasureInCentimeters();

// Detection Obstacle via Telemetres
if( ((TelGauche > Min_Obstacle) && (TelGauche <=
Detection_obstacleMax)) || ((TelAvant > Min_Obstacle) && (TelAvant <=
Detection_obstacleMaxAvant)) || ((TelDroite > Min_Obstacle) &&
(TelDroite <= Detection_obstacleMax)) ){
    switch (detecterObstacle(TelGauche, TelAvant, TelDroite)) {
        case 1: // Detection : TelDroite OU TelAvant
            Serial.println("TOURNER A GAUCHE");
            delay(200);
            break;
        case 2: // Detection : TelGauche
            Serial.println("TOURNER A DROITE");
            delay(200);
            break;
        case 3: // Detection : TelGauche ET TelDroite
            Serial.println("AVANT");
            delay(200);
            break;
    } // Fin du Switch Case
} // Fin du if Telemetre
} // Fin du loop

// FONCTIONS
int detecterObstacle(unsigned long distGauche, unsigned long distAvant,
unsigned long distDroite) {
    if( ((distGauche > Min_Obstacle) && (distGauche >
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        return 1; // Obstacle Avant + Droite
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
> Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        return 2; // Obstacle GAuche + Avant
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche >
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
> Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        return 2; // Avant seul - Tourner a Droite
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite

```

```
> Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
> Detection_obstacleMaxAvant)) ){
    return 2; // Obstacle que a Gauche - Tourner a Droite
}
else if( ((distGauche > Min_Obstacle) && (distGauche >
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
> Detection_obstacleMaxAvant)) ){
    return 1; // Obstacle que a Droite - Tourner a Gauche
}
else if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
> Detection_obstacleMaxAvant)) ){
    return 3; // Gauche + droite = Avancer droit
}
} // FIN DE LA FONCTION
```

● Le Capteur de couleur :

Pour ce programme, nous faisons la lecture du sol via code RGB, permettant par la suite de savoir si c'est le sol est bleu, alors le robot avant car il est sur la piste, et si c'est un seuil blanc, alors le robot s'arrete car il est sur la zone finale.

```
#include "Adafruit_TCS34725.h"

Adafruit_TCS34725 tcs =
Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

// CONFIGURATION
void setup(){
  Serial.begin(2400);
  tcs.begin();

  pinMode(13, OUTPUT); // Met la LED du Capteur en sortie
  digitalWrite(13, LOW); // Eteins la LED du capteur
}

// PROGRAMME PRINCIPAL
void loop() {
  uint16_t r, g, b, c;
  tcs.getRawData(&r, &g, &b, &c);

  // Lit la valeur en decimal du code RGB
  Serial.print("R: "); Serial.print(r, DEC); Serial.print(" ");
  Serial.print("G: "); Serial.print(g, DEC); Serial.print(" ");
  Serial.print("B: "); Serial.print(b, DEC); Serial.print(" ");
  Serial.println(" ");
}
```

- **L'émission Infrarouge (Balise) :**

Pour ce programme, nous envoyons un signal qui est une trame (modulé automatiquement via la fonction), envoyant un mot en Hexadécimal, permettant ainsi à notre robot de lire que cette valeur, nous pourrons ensuite la modifier à notre guise si nous savons qu'une autre balise envoie le même mot. Ici il s'agit du mot : FINAL

```
#include "IRremote.h"

IRsend irsend = 2; // Envoie le signal via le pin 2

void setup() {
}

void loop () {
    // Envoie en répétition du signal (mot FINAL en hexa)
    for (int i = 0; i < 3; i++) {
        irsend.sendNEC(0xBFD11, 8); // BONNE VALEUR A DÉTECTOR
        delay(50);
    }
    delay(100);
}
```

- **La réception Infrarouge :**

Pour ce programme, nous récupérons le signal qui nous est envoyé par l'émission, que l'on reçoit en décimal, nous testons donc chaque récepteur en vérifiant à chaque que si nos récepteur reçoit un signal, que ce soit NOTRE mot, permettant ainsi de limiter chaque réception et avoir des réponses simples.

Nous pourrons par la suite, dire si on détecte à l'avant (qui est prioritaire) on avance, si on détecte à droite via récepteur droite, on tourne vers la balise donc à droite et pareil pour la gauche.

```
// A mettre dans le robot - Récepteur
#include <IRremote.h>
#include <avr/io.h>

// Dans chaque code de balises : base converti 32
#define BaliseFinal 6914112 // Valeur à comparer de l'émission
// Balise Final - F:6 I:9 N:14 A:1 L:12

IRrecv irrecv_Gauche(9); // Pin entrée Infrarouge Reception Gauche
IRrecv irrecv_Avant(8); // Pin entrée Infrarouge Reception Avant
IRrecv irrecv_Droite(3); // Pin entrée Infrarouge Reception Droite
decode_results results;
```

```
// CONFIGURATION
void setup() {
    Serial.begin(2400);

    // Configure les récepteurs
    irrecv_Gauche.enableIRIn(); // Infrarouge Reception - Gauche
    irrecv_Avant.enableIRIn(); // Infrarouge Reception - Avant
    irrecv_Droite.enableIRIn(); // Infrarouge Reception - Droite
}

// PROGRAMME PRINCIPAL
void loop() {
    if ((irrecv_Avant.decode(&results)) && (results.value ==
BaliseFinal)) {
        Serial.println("Avant : "); Serial.println(results.value);
    }
    else if((irrecv_Gauche.decode(&results)) && (results.value ==
BaliseFinal)){
        Serial.println("Gauche : "); Serial.println(results.value);
    }
    else if((irrecv_Droite.decode(&results)) && (results.value ==
BaliseFinal)) {
        Serial.print("Droite : "); Serial.println(results.value);
    }
    else {
        Serial.print("Autre / rien : "); Serial.println(results.value);
    }

    // Prépare le récepteur pour la prochaine réception - ACTUALISATION
    irrecv_Gauche.resume();
    irrecv_Avant.resume();
    irrecv_Droite.resume();

    Serial.println("-----");
    delay(500);
}
```

- **Le servomoteur (Éclatement Ballon) :**

Pour ce programme, il s'agit d'uniquement faire appel à une fonction en dehors du main, permettant de faire la rotation de notre servo moteur, nous faisons cette fonction plusieurs fois afin d'être certain que le ballon éclate comme il faut.

```
#include <Servo.h>

Servo servo_A4;
int test = 0; // Eclate le Ballon plusieurs fois

// CONFIGURATION
void setup() {
    Serial.begin(9600);
    servo_A4.attach(A4);
}

// PROGRAMME PRINCIPAL
void loop() {
// execution 3 fois pour que le ballon soit sur d'etre exploser
    for(; test < 3; test++){
        ExplosionBallon_loop(); // Appel fonction eclate ballon
    }
    delay(100);
}

// FONCTIONS
int ExplosionBallon_loop(){
    servo_A4.write(90); // Ballon eclate, retour position depart
    delay(1000);

    servo_A4.write(0); // Ballon eclate, retour position depart
    delay(1000);
}
```

- **Le démarrage Jack :**

Pour ce programme, nous faisons uniquement la lecture de l'entrée de notre PIN, permettant ainsi que si notre jack est branché, nous sommes donc au départ, notre robot reste à l'arrêt, puis quand nous retirerons le jack, notre robot se lancera directement, lisant une valeur nulle.

```
// CONFIGURATION
void setup() {
    Serial.begin(9600);
    pinMode(A5, INPUT);
}

// PROGRAMME PRINCIPAL
void loop() {
    int Etat;

    Etat = digitalRead(PIN); // Etat du Pin : 1 = branche / 0 = debranche
    Serial.println(Etat);

    delay (100);
}
```

- **Le programme final (union de toutes les fonctions) :**

Attribution des PINs pour les composants via la carte Arduino					
0	Non utilisé (RX)	7	Sens Moteur Droite	A0	Non utilisé
1	Non utilisé (TX)	8	Récepteur IR Avant	A1	Non utilisé
2	Servomoteur	9	Récepteur IR Droite	A2	Non utilisé
3	Récepteur IR Gauche	10	Non utilisé	A3	Entrée Jack
4	Sens Moteur Gauche	11	Télémètre Droite	A4	SDA (Capteur couleur)
5	Moteur Gauche	12	Télémètre Avant	A5	SCL (Capteur Couleur)

6	Moteur Droite	13	Télémètre Gauche		
SDA	HS	SCL	HS		

- **Programme Final :**

Ici nous faisons tout simplement l'union de chaque fonction tout en l'adaptant au fur et à mesure selon les conditions, par exemple en règle général il s'agirait des limites de détection de nos télémètres ou alors la vitesse de nos moteurs.

```
// Declaration des bibliothèques
#include <IRremote.h>
#include <avr/io.h>
#include <Wire.h>
#include "Ultrasonic.h"
#include "Adafruit_TCS34725.h"
#include <Servo.h>
#include <NewPing.h>

// Configuration du capteur de couleur
Adafruit_TCS34725 tcs =
Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X);

// Configuration des Telemetres
Ultrasonic ultrasonic1(13);
Ultrasonic ultrasonic2(12);
Ultrasonic ultrasonic3(11);

#define Min_Obstacle 2
#define Detection_obstacleMax 17
#define Detection_obstacleMaxAvant 30

// Configuration Servo Moteur
Servo servo_2;
int test = 0; // Permet de faire la boucle pour repeter l'explosion du ballon

// Configuration des roues
#define Vitesse_Roues 135 // Vitesse moteur
#define Vitesse_Esrieve 100 // Vitesse moteur Virage
#define Vitesse_Arret 0      // Vitesse d'arrêt
```

```
int Moteur_Gauche = 5; // M1+
int Sens_Gauche = 4; // S1
int Moteur_Droite = 6; // M2+
int Sens_Droite = 7; // S2

// Configuration Balise
#define BaliseFinal 6914112 // Valeur à comparer de l'émission - Balise
Final - F:6 I:9 N:14 A:1 L:12

// Config Infrarouge Reception
IRrecv irrecv_Gauche(9);
IRrecv irrecv_Avant(8);
IRrecv irrecv_Droite(3);
decode_results results;

// Autres Variables
int Depart = 1; // Permet de quitter la zone blanche de départ
int Etat; // Permet la lecture du prise jack

// CONFIGURATION - Pin d'entrées et de sorties
void setup() {
    Serial.begin(2400);
    pinMode(A3, INPUT);
    tcs.begin();
    servo_2.attach(2);

    irrecv_Gauche.enableIRIn(); // Infrarouge Reception - Gauche
    irrecv_Avant.enableIRIn(); // Infrarouge Reception - Avant
    irrecv_Droite.enableIRIn(); // Infrarouge Reception - Droite

    pinMode(Moteur_Gauche, OUTPUT);
    pinMode(Sens_Gauche, OUTPUT);
    pinMode(Moteur_Droite, OUTPUT);
    pinMode(Sens_Droite, OUTPUT);
}

// PROGRAMME PRINCIPAL
void loop() {
// Lecture des distances via télemètres
    long TelGauche = ultrasonic1.MeasureInCentimeters();
    long TelAvant = ultrasonic2.MeasureInCentimeters();
    long TelDroite = ultrasonic3.MeasureInCentimeters();

    Etat = digitalRead(A3); // lit l'état du pin du jack}
```

```

if(Etat == 0){ // si jack a 0 -> Jack retire donc robot roule
    if(Depart == 1){ // Quitter la zone de départ
        Avancer();
        delay(2000);
        Depart = 0; // eviter de retourner dans la condition
    }

    // Lit les couleurs du capteurs du couleur
    uint16_t r, g, b, c;
    tcs.getRawData(&r, &g, &b, &c);

    // Color Bleu - donc sur la piste on roule
    if((r < 500) && (g < 500) & (b < 500)) {
        // Detection Obstacle via Telemetres
        if( ((TelGauche > Min_Obstacle) && (TelGauche <=
Detection_obstacleMax)) || ((TelAvant > Min_Obstacle) && (TelAvant <=
Detection_obstacleMaxAvant)) || ((TelDroite > Min_Obstacle) &&
(TelDroite <= Detection_obstacleMax)) ){
            // Balise a Gauche + obstacle devant -> direction la balise
            if( ((irrecv_Gauche.decode(&results)) && (results.value ==
BaliseFinal)) && ((TelAvant > Min_Obstacle) && (TelAvant <=
Detection_obstacleMaxAvant)) ) {
                GaucheBalise();
                delay(300);
                Avancer();
                delay(200);
            }
            else {
                // Fonctions des telemetres
                switch (detecterObstacle(TelGauche, TelAvant, TelDroite)) {
                    case 1: // Detection : TelDroite
                        GaucheBalise();
                        delay(300);
                        Avancer();
                        delay(200);
                        break;
                    case 2: // Detection : TelDroite + TelAvant
                        DroiteBalise();
                        delay(300);
                        Avancer();
                        delay(200);
                        break;
                    case 3:
                        Avancer();
                        delay(500);
                        break;
                    case 4:
                        Reculer();
                        break;
                }
            }
        }
    }
}

```

```

        } // fin du switch case
    } // fin du else pour telemetres
} // Fin du if Genereal des telemetres

// Balise FINAL
// Si detection balise a l'avant
if ((irrecv_Avant.decode(&results)) && (results.value ==
BaliseFinal)) {
    Serial.println("AvantFinal : "); Serial.println(results.value);
    Avancer();
}
// Sinon detecte a gauche
else if((irrecv_Gauche.decode(&results)) && (results.value ==
BaliseFinal)){
    Serial.println("GaucheFinal : ");
    Serial.println(results.value);
    GaucheBalise();
    delay(300);
}
// Sinon si detecte a droite
else if((irrecv_Droite.decode(&results)) && (results.value ==
BaliseFinal)) {
    Serial.print("DroiteFinal : "); Serial.println(results.value);
    DroiteBalise();
    delay(300);
}

// Pas de balise NI telemetres on avance jusqu'a detecter quelque chose
else {
    Serial.print("RIEN : "); Serial.println(results.value);
    Avancer();
}
} // Sol de couleur blanche = zone final
else {
    Serial.println("Blanc");
    Stop(); // Assure de stoper le robot
    delay(500);

    // Execution 3 fois pour que le ballon soit sur d'etre Eclate
    for(; test < 3; test++){
        ExplosionBallon_loop();
    }
} // fin du else pour couleur sol
} // etat jack a 1 donc stop
else {
    Stop();
}

```

```
// Prépare le récepteur pour la prochaine réception
irrecv_Gauche.resume();
irrecv_Avant.resume();
irrecv_Droite.resume();

delay(250);
} // fin du programme (loop)

// FONCTIONS
// Déplacement
void Avancer() {
    digitalWrite(Sens_Gauche, 1);
    digitalWrite(Sens_Droite, 1);

    analogWrite(Moteur_Gauche, Vitesse_Roues - 5);
    analogWrite(Moteur_Droite, Vitesse_Roues + 22);
}

void Stop(){
    digitalWrite(Sens_Gauche, 1);
    digitalWrite(Sens_Droite, 1);

    analogWrite(Moteur_Gauche, 0);
    analogWrite(Moteur_Droite, 0);
}

void Reculer() {
    digitalWrite(Sens_Gauche, 1);
    digitalWrite(Sens_Droite, 1);

    analogWrite(Moteur_Gauche, 0);
    analogWrite(Moteur_Droite, 0);
    delay(1000);

    digitalWrite(Sens_Gauche, 0);
    digitalWrite(Sens_Droite, 0);

    analogWrite(Moteur_Gauche, Vitesse_Roues - 5);
    analogWrite(Moteur_Droite, Vitesse_Roues + 22);
}

void GaucheBalise() {
    digitalWrite(Sens_Gauche, 0);
    digitalWrite(Sens_Droite, 1);

    analogWrite(Moteur_Gauche, 90);
    analogWrite(Moteur_Droite, Vitesse_Esquive + 70);
}
```

```

void DroiteBalise() {
    digitalWrite(Sens_Gauche, 1);
    digitalWrite(Sens_Droite, 0);

    analogWrite(Moteur_Gauche, Vitesse_Esrieve + 70);
    analogWrite(Moteur_Droite, 90);
}

// Détection via telemetres
int detecterObstacle(unsigned long distGauche, unsigned long distAvant,
unsigned long distDroite) {
    if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        Serial.println("Gauche + Avant + Droite");
        return 4; // Obstacle GAuche + Avant + droite
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche >
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        Serial.println("DROITE + Avant");
        return 1; // Obstacle Avant + Droite
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
> Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        Serial.println("Gauche + Avant");
        return 2; // Obstacle GAuche + Avant
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche >
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
> Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
<= Detection_obstacleMaxAvant)) ){
        Serial.println("AVANT");
        return 2; // Avant seul - Tourner a Droite
    }
    else if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
> Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
> Detection_obstacleMaxAvant)) ){
        Serial.println("GAUCHE");
        return 2; // Obstacle que a Gauche - Tourner a Droite
    }
}

```

```
else if( ((distGauche > Min_Obstacle) && (distGauche >
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
> Detection_obstacleMaxAvant)) ){
    Serial.println("DROITE");
    return 1; // Obstacle que a Droite - Tourner a Gauche
}
else if( ((distGauche > Min_Obstacle) && (distGauche <=
Detection_obstacleMax)) && ((distDroite > Min_Obstacle) && (distDroite
<= Detection_obstacleMax)) && ((distAvant > Min_Obstacle) && (distAvant
> Detection_obstacleMaxAvant)) ){
    Serial.println("Gauche + Droite");
    return 3; // Gauche + droite = Avancer droit
}
}

// Explosion du ballon = zone finale
int ExplosionBallon_loop(){
    servo_2.write(0); // Ballon eclate, retour position depart
    delay(1000);

    servo_2.write(90); // Ballon eclate, retour position depart
    delay(1000);
}
```

Jours du concours

● Jeudi Après-midi :

En arrivant à Cachan et en voulant faire des essais avec le robot, nous avons réalisé que les 3 télémètres (HC-SR04) ne fonctionnaient plus. Pour palier ce problème, nous avons voulu en ressouder 3 autres que nous avions en double mais le fer à souder que nous avons pris ne fonctionnait pas (il ne chauffait pas). Alors, pour pouvoir faire nos tests en attendant de récupérer un fer à souder fonctionnel, nous avions une autre référence de télémètres que nous avions en double dans notre boîte, ceux-ci ne nécessitant pas de soudure pour être cablés, ils avaient déjà leur câbles.

Une fois le soucis des télémètres réglé, nous avons pu effectuer quelques essais sur piste, mais cette fois-ci c'était un problème au niveau de la détection des balises qui se présentait à nous, en faisant des essais, nous avons découvert que la balise utilisant la résistance de puissance ainsi que les 3 LEDs était bien mieux détectée par notre robot que les 2 autres balises avec seulement 2 LEDs.

Nous avons donc pour le reste des jours décidé d'utiliser la balise à 3 LEDs. Ce jour-là, nous n'avons pas passé d'homologation.

● Vendredi Matin :

Afin que les télémètres tiennent mieux et détectent correctement, vu que nous avions un autre fer à souder, nous avons décidé de souder 3 télémètres HC-SR04 (nos télémètres de base), les autres télémètres ne rentraient pas dans la conception 3D.

Une fois cela fait et quelques essais sur piste, nous avons pu remarquer que le capteur de couleur avait du mal à différencier le bleu du blanc, nous avons essayé de l'approcher du sol car on se disait que son problème de détection venait sûrement du fait de son éloignement avec le sol.

Mais cela n'a pas résolu notre problème et donc le robot ne pouvant pas s'arrêter sur la zone blanche de fin, il ne pouvait pas être homologué, nous n'avons donc pas tenté d'homologation.

- **Vendredi Après-midi :**

Sur cette demi-journée, nous avons pu découvrir que le problème du capteur de couleur ne venait pas de son positionnement par rapport au sol, mais en fait du code utilisé. Le code utilisé n'était pas bon et ne permettait pas de bien différencier les couleurs, (l'écart de proportion de bleu entre le bleu et le blanc était trop faible).

C'est donc en utilisant un autre code exemple pour le capteur de couleur et en le réadaptant au code complet que le robot pouvait correctement faire la différence entre le bleu et le blanc, l'écart de proportion de bleu entre le bleu et le blanc était bien plus important. Il restait quand même un problème à nos yeux, le robot avait vraiment du mal à détecter la balise à l'avant malgré le fait qu'on utilisait la balise à 3 LEDs.

Nous avons en fait remarqué que le capteur infrarouge central était plus bas que les autres et ne pouvait donc qu'à moitié détecter, l'autre partie pour la détection étant camouflée par le plastique du haut du robot. Pour résoudre ce problème, nous avons donc dessoudé le photo transistor central et nous en avons soudé un autre plus haut afin qu'il puisse pleinement détecter la balise.

Après cela nous avons pu faire quelques essais qui étaient concluants pour une future homologation.

- **Samedi Matin :**

Durant cette matinée, nous avons voulu retester le robot pour la future homologation, mais il avait à nouveau du mal à détecter la balise, après avoir vérifié plusieurs fois le branchement des différents composants pour voir si ça ne venait pas de là, nous avons en fait remarqué que c'était la pile qui faiblissait, en la changeant, le robot était à nouveau fonctionnel. Une fois cette pile changée, nous avons pu passer l'homologation qui consistait à parcourir la diagonale sans obstacle et s'arrêter sur la zone blanche.

L'homologation étant réussie, nous avons pu participé à une première épreuve, sur la 1ère manche, le robot s'est pris le 1er obstacle, nous avons donc marqué 0 points sur cette manche, la seconde manche, nous avons donc décidé de retirer le robot directement de la zone de départ.

Après cela, nous avons participé à un second parcours, pour celui-ci, nous avons décidé lors des 2 manches de retirer le robot directement, nous avons ainsi pu marquer 2 points car personne n'était arrivé dans sa zone blanche. Nous permettant d'accéder à la finale.

- **Samedi Après-midi :**

Après cela nous avons décidé d'effectuer des essais sur le robot pour améliorer son évitement d'obstacle. Avec l'aide d'Abderrahim, nous avons pu voir qu'il manquait des conditions pour les évitements d'obstacles, avec ces modifications, le robot détectait mieux les obstacles et donc les évitait mieux.

Après cela, nous avons décidé en voyant que le robot avait toujours des angles morts de remplacer les télémètres en repassant sur ceux qui ont déjà des câbles et en les rapprochant, celà a permis au robot de mieux détecter les obstacles.

Lors de la finale, le robot une fois lancé s'est mis à tourner autour d'un obstacle comme s'il ne détectait plus la balise. Nous n'avons donc pas marqué de points.

Durant la 2ème manche, nous avons donc remplacé la pile, pensant que le problème pouvait venir de là, mais la pile était vide ou défectueuse, le robot ne s'est donc pas élancé, nous n'avons donc pas marqué de points.

Résultat

Après plusieurs jours de réglages intensifs et de résolution de problèmes, nous avons enfin réussi à homologuer notre robot le dernier jour de la compétition. Grâce à notre persévérance, nous avons pu participer à deux épreuves avant la finale.

Lors de ces épreuves, nous avons pu obtenir un total de 2 pauvres points, par le fait que nous avions enlevé notre robot et qu'aucun robot adverse n'avait réussi à atteindre sa zone d'arrivée, mais cela nous a permis malgré tout de passer à la phase finale n'ayant pas eu d'autres robots inférieur dépassant notre score.

(Tout au long du concours, nous avons pu échanger avec diverses équipes, et la grande majorité avait les mêmes problèmes que nous “L'électronique”)

Finalement, nous avons pu atteindre la 4eme place.

Ayant rencontré des problèmes durant ce parcours qui était : La pile, nous avions fait l'utilisation d'une pile déchargé et c'est pour cette raison que notre robot perdait tout signal de notre balise mais aussi, notre robot se trouvait en sous alimentation pour l'entièreté de nos composant, l'empêchant ainsi de pouvoir faire la détection d'obstacle ou aussi le suivi de notre signal infrarouge.

Lors de la seconde tentative durant cette finale, le robot ne démarrait même plus, prouvant le manque d'énergie dans notre pile, ayant ensuite fait le test hors concours avec une pile neuve, notre robot marchait parfaitement bien, prouvant ainsi notre manque de préparation où l'on aurait dû faire l'acquisition de plusieurs piles et batteries neuves.

Retour sur Expérience

- **Matthieu MOCHET :**

Ce concours m'a permis d'apprendre à utiliser un logiciel de modélisation 3D (fusion 360) en partant de zéro, c'est un logiciel que j'ai apprécié apprendre à l'utiliser. Il me reste beaucoup de choses à apprendre sur ce logiciel, beaucoup de choses que je ne maîtrise pas encore et il m'a aussi permis de me rendre compte que vouloir faire un robot trop compliqué c'est parfois pas du tout arrangeant pour l'utilisation qu'on en fait au final.

Ce concours m'a aussi permis de travailler sur la réalisation de typons pour différents systèmes, la réception et l'émission infrarouge. J'ai aussi beaucoup travaillé sur la soudure de composants. Ce concours m'a beaucoup apporté sur le travail en équipe et la répartition des tâches.

- **Aubin TOURAIS :**

En ce qui me concerne, ce concours a été une source d'émotions du début à la fin, avec différents obstacles qui m'ont fait barrages que ce soit en programmation mais surtout en électronique, quand nous voyons que lors du jour de la présentation, nous partions mal avec la réussite de l'homologation au tout dernier moment le samedi matin, puis quand on nous apprend que nous atteignons la finale de justesse, rien qu'à ce moment là, la reprise en confiance en soit était de retour.

Il y a malgré tout une déception importante en soit que je garde quand même, quand on sait que nous pouvions faire beaucoup mieux, rien qu'en réussissant plusieurs fois le parcours final alors que nous n'avions réussi aucune autre suite à de nombreuses erreurs de calibration ou encore de fonctions se trouvant dans le code. Si nous avions trouvé les solutions bien plus tôt, la réussite mais surtout un meilleur résultat en termes de points se serait ouvert à nous.

Néanmoins, je garde cela comme une source d'expérience vraiment plaisante, avec le fait que nous avons réussi à trouver de nombreuses soluces comme avec l'infrarouge, ou encore le choix de la stratégie.

Si l'occasion se rouvrait à moi, je n'hésiterai pas à prendre ma revanche, en gardant la même stratégie mais surtout en modifiant le robot, ne partant pas sur les mêmes bases et en changeant l'intégralité des méthodes, comme l'utilisation du Raspberry, d'un Lidar mais aussi d'un robot plus simple à manipuler.

Durant ce concours, ce que j'ai moins aimé étant sans aucun doute l'électronique, avec les nombreux problèmes que nous avons rencontré et surtout l'histoire avec les télémètres, dont je ne comprend toujours pas la raison...

Ce concours m'a prouvé encore une fois que le challenge est toujours là, que l'amour pour la programmation mais aussi la robotique est ce que j'aime, cela m'a bien montré que cette formation du BUT GEII était ce que je voulais vraiment suivre pour mes études.

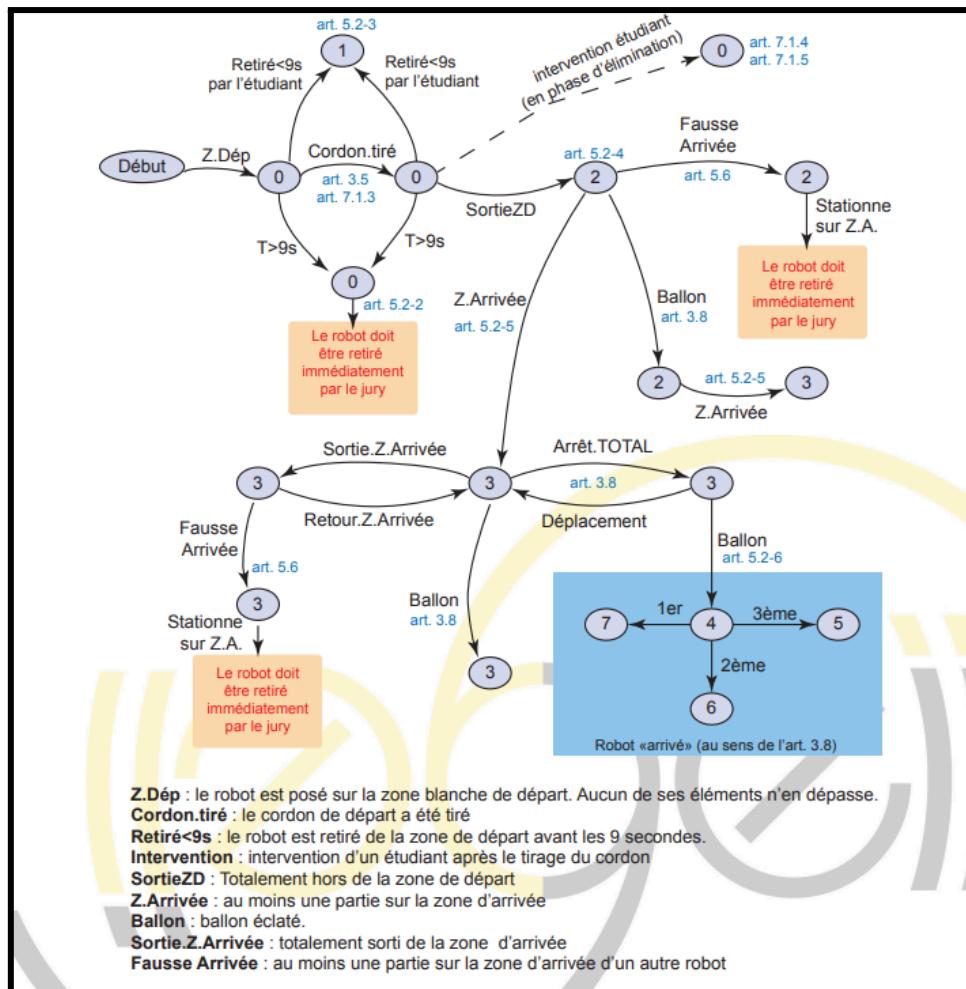
Conseils et remarques

- Pour la conception du robot, faire un robot simple, le fait d'avoir une coque fermée faisait qu'il fallait parfois démonter le robot et en enlevant cette coque, certains câbles pouvaient par inadvertance se déconnecter, provoquant des problèmes sur le robot. Faire un robot ouvert avec des "étages" accessibles permet de réduire les risques et de voir plus facilement si quelque chose est débranché.
- En utilisant plusieurs télémètres, il faut prendre en compte leur faible angle d'émission dans la conception et ne pas trop les éloigner, sinon ils auront des angles morts (entre 2 télémètres) et le robot pourra se cogner contre des obstacles car il ne les captera pas avec ses télémètres.
- Noter les piles qui sont vides et celles qui sont pleines, pour éviter d'utiliser le robot avec une pile vide.
- Pour la détection d'obstacle et la cartographie du parcours, utiliser un capteur Lidar.
- Ne pas faire un robot fermé avec les composants à l'intérieur, le fait de "démonter" ce dernier peut débrancher les fils et donc faire des problèmes avec le robot.
- Réaliser des typons pour le robot et ne pas utiliser de plaques à trous, en utilisant des typons, le fait de souder apportera de la stabilité aux câbles et ainsi ils auront moins de chance de se débrancher lorsque le robot roule.
- Plutôt que d'utiliser plusieurs phototransistor pour détecter la balise, en utiliser qu'un seul mais le faire balayer sur une certaine plage via un servo moteur.
- Lorsqu'il faut ajuster le câblage du robot, bien penser à débrancher la carte arduino de son alimentation (pile ou ordinateur) pour éviter qu'un composant ne lâche, mais aussi penser à débrancher la batterie, même si cette dernière n'est censé alimenté

que le shield et les moteurs, il est possible qu'elle fasse un retour de courant sur les composants et les rendent défectueux.

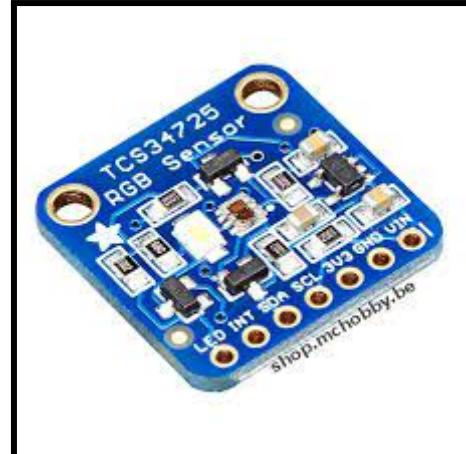
- Utiliser plusieurs cartes Arduino, 1 pour les moteurs, 1 pour les capteurs/entrées (Infrarouges, télémètres, couleur, Jack, arrêt d'urgence, encodeurs) et 1 carte qui communique entre les 2 cartes pour faire le lien.
- Faire l'utilisation d'un Lidar afin de pouvoir visualiser l'entièreté de la zone et pouvoir ainsi mieux détecter les différents obstacles, les murs.
- Utiliser une carte Raspberry pouvant ainsi avoir plus de dissipation de courant dans les différents composant étant donné que la carte Arduino est elle beaucoup trop limité (la carte arduino ne suffit pas pour un Lidar par exemple), faisant ainsi la possible propagation de chute de tension ce qui nous a causé la perte de nombreux télémètres.
- Pour le jack, repartir sur un neuf, le Jack actuel lâche beaucoup trop facilement.

Annexes



Annexe A.1 : Diagramme d'état définissant le comptage des points : Sujet Concours

CACHAN



Annexe B.1 : *Capteur de couleur RGB - TCS34725* : Site MCHobby



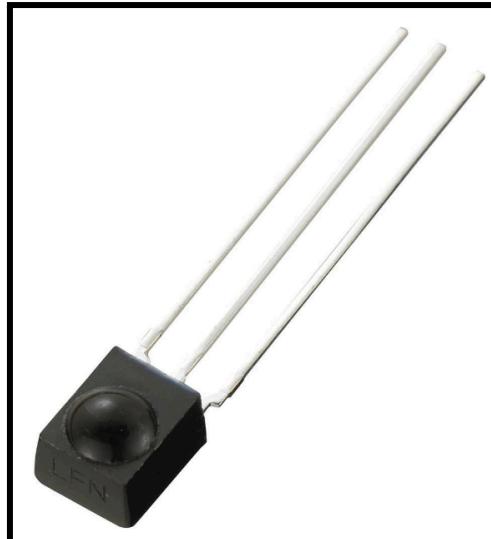
Annexe B.2 : *Télémètre à ultrason - HC-SR04* : Site GoTronic



Annexe B.3 : *Ensemble kit roue + Motoréducteur + Support 19E837RK* : Site Lextronic



Annexe B.4 : *Emetteur Infrarouge* : Site Lextronic



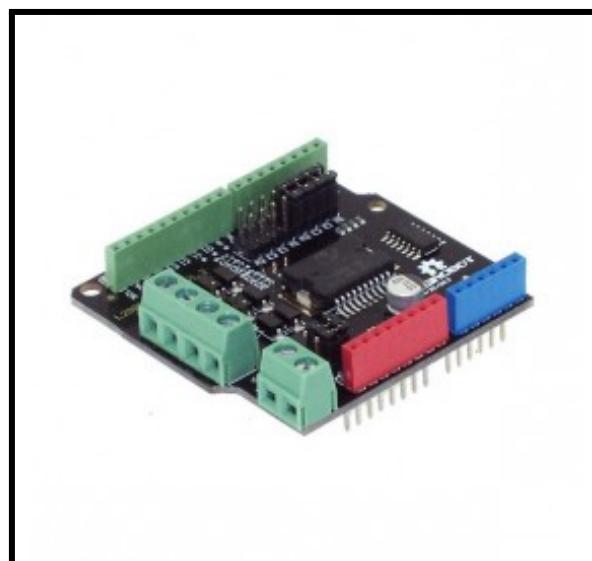
Annexe B.5 : *Récepteur Infrarouge* : Site Conrad



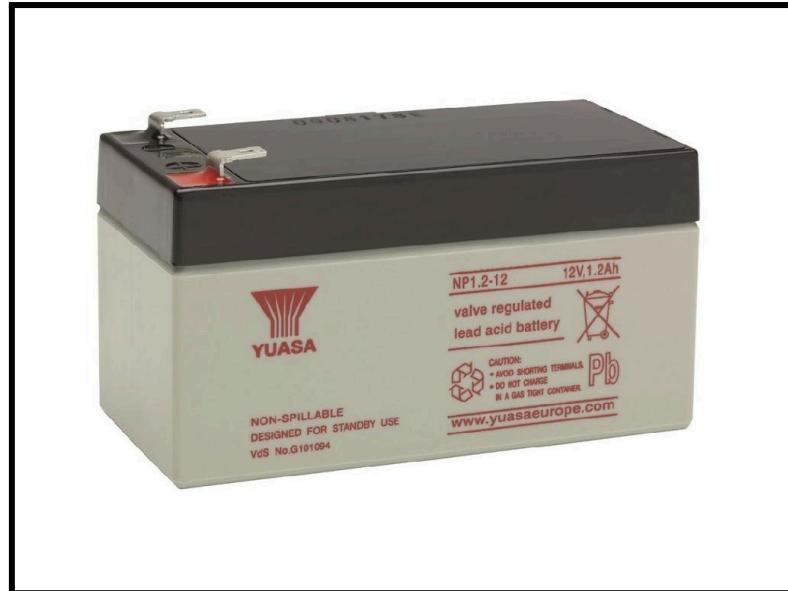
Annexe B.6 : *Servomoteur - S-0009 MG* : Site Conrad



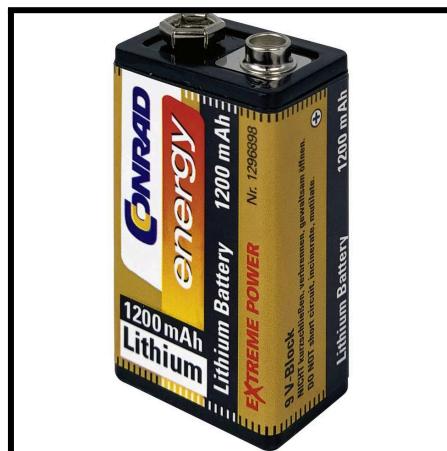
Annexe B.7 : *Carte Arduino UNO A000066* : Site GoTronic



Annexe B.8: *Shield - L298p shield v1.3*: Site GoTronic



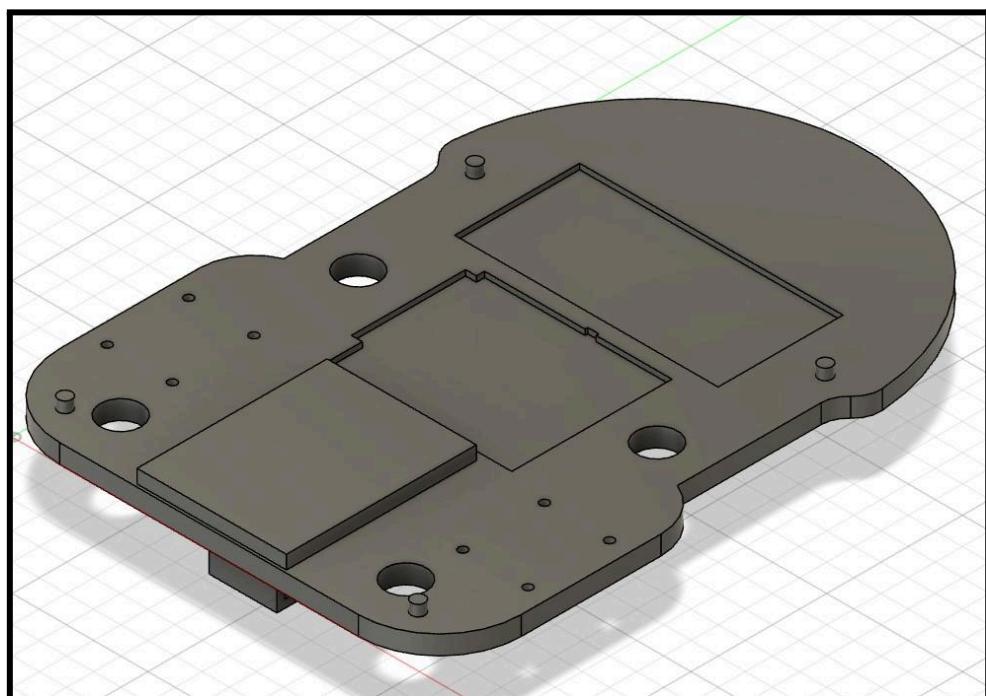
Annexe B.9 : Batterie 12 V - NP1.2-12 : Site YUASA



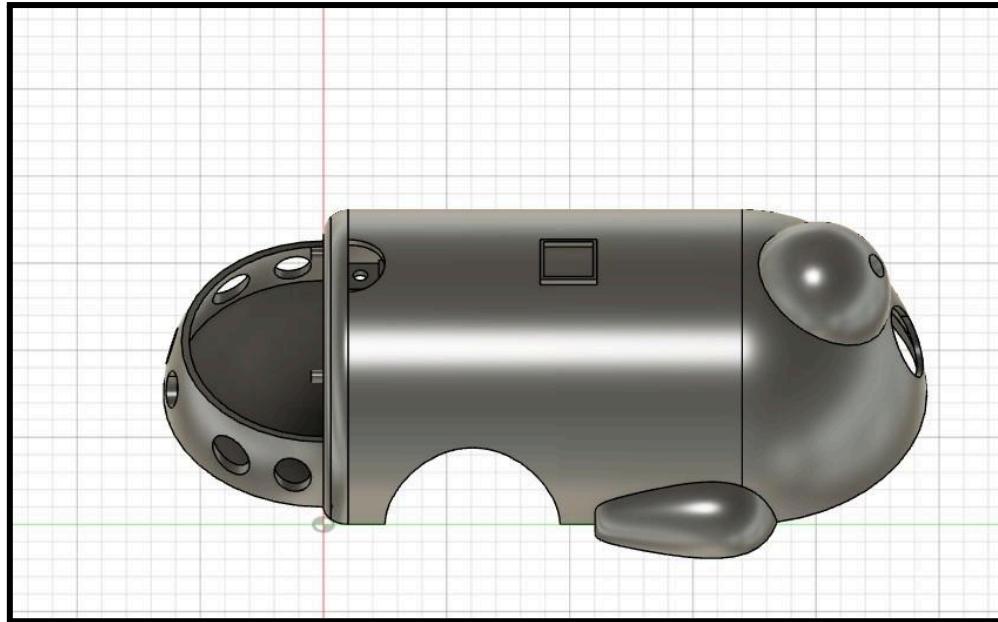
Annexe B.10 : Pile 9 V - Conrad Energy Extreme Power 6LR61 Pile 6LR61 (9 V) lithium
1200 mAh : Site Conrad



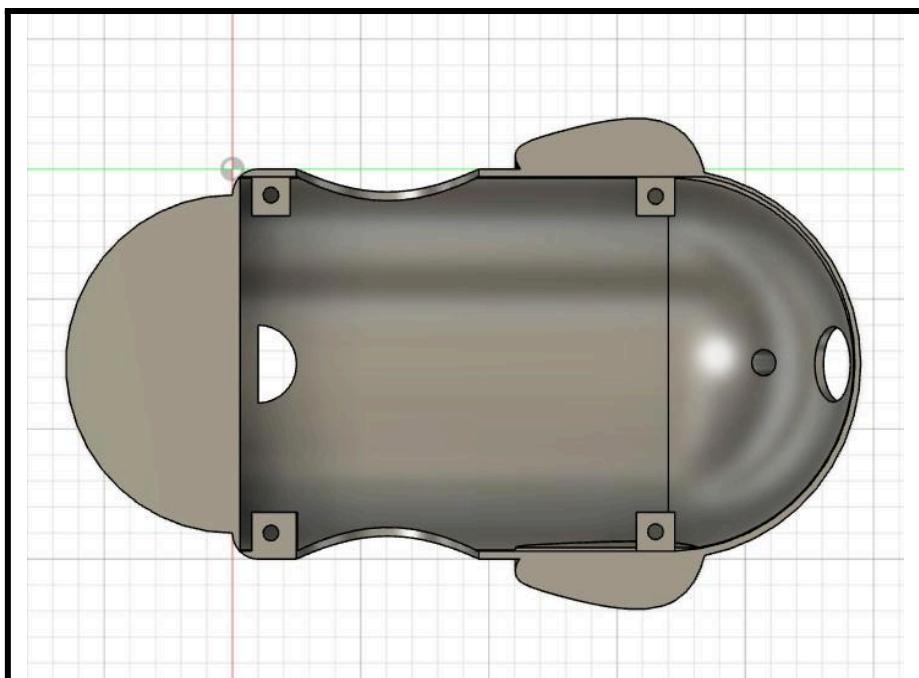
Annexe B.11 : *Clip de batterie 9V avec prise jack 5,5 mm* : Site GoTronic



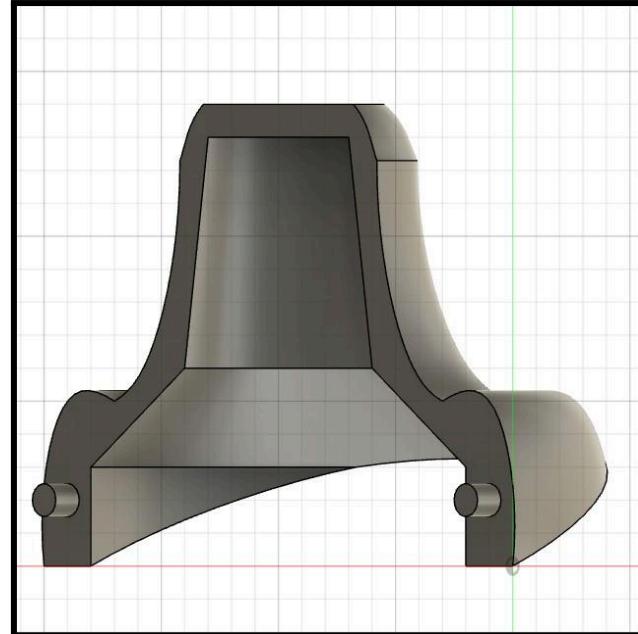
Annexe C.1 : *Conception du châssis de notre robot (lapin)* : Équipe IUT Evry



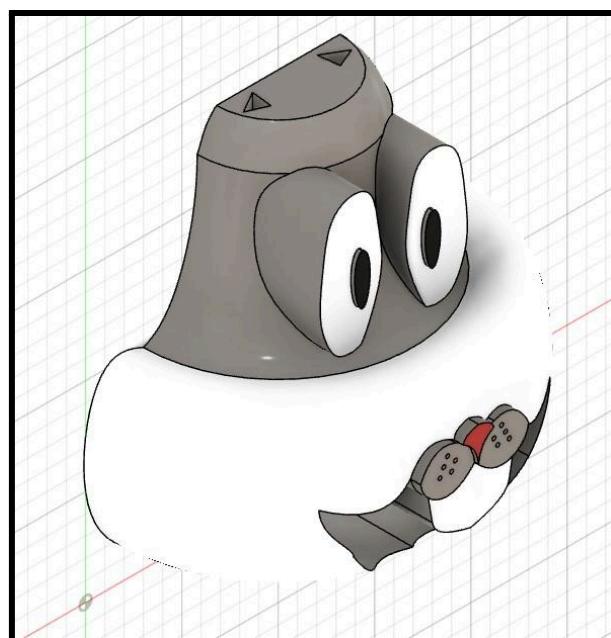
Annexe C.2 : *Conception du dos de notre robot (lapin)* : Équipe IUT Evry



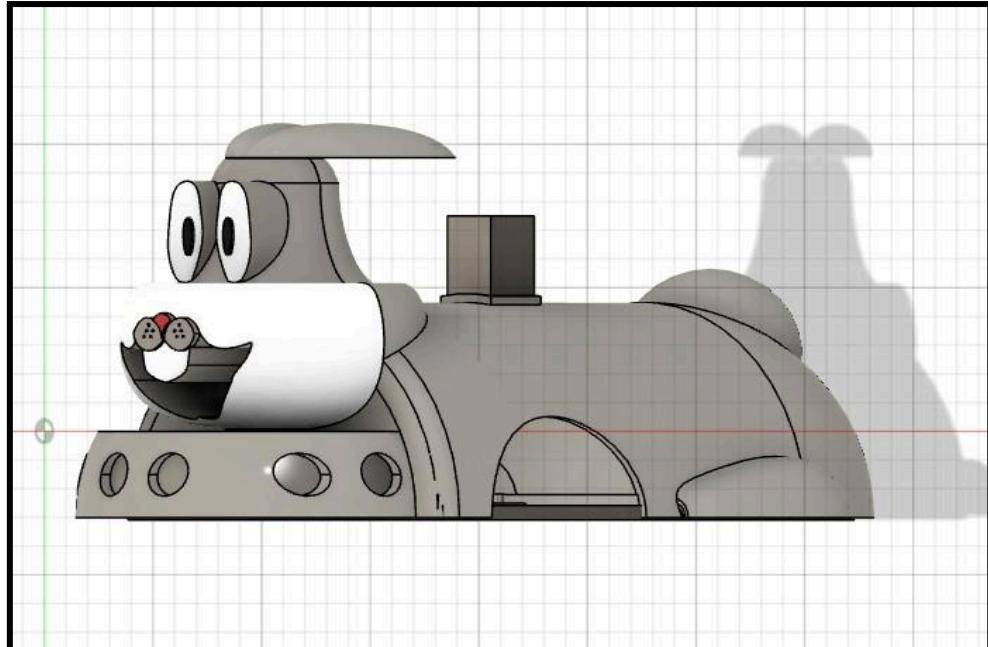
Annexe C.3: *Conception des fixations pour le dos de notre robot (lapin)* : Équipe IUT Evry



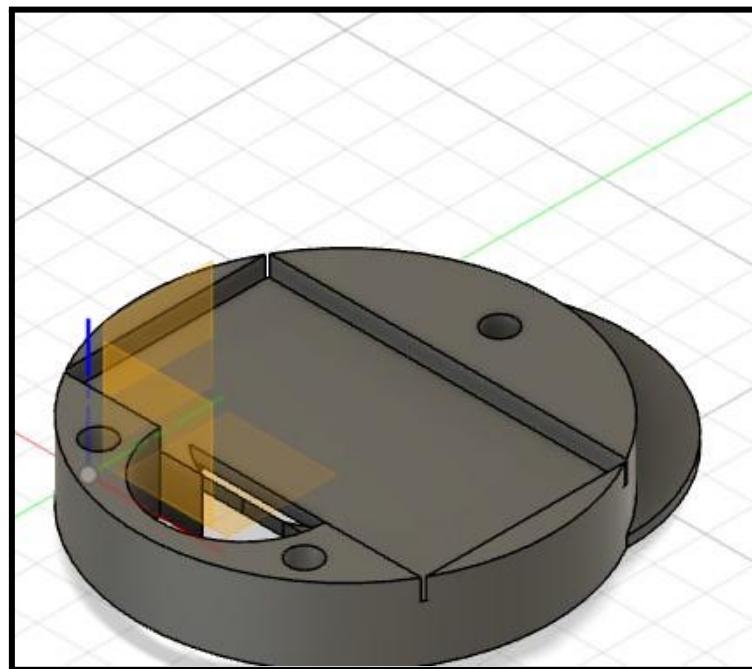
Annexe C.4 : Conception arrière de la tête du lapin : Équipe IUT Evry



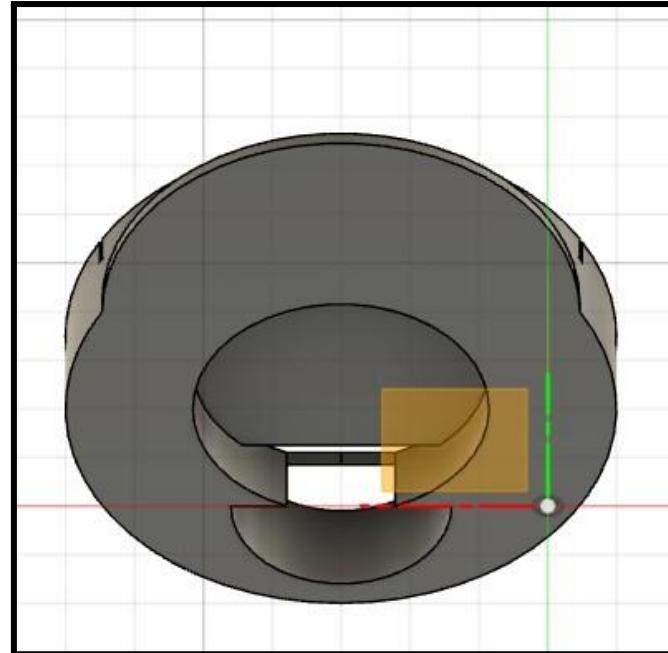
Annexe C.5 : Conception avant de la tête du lapin : Équipe IUT Evry



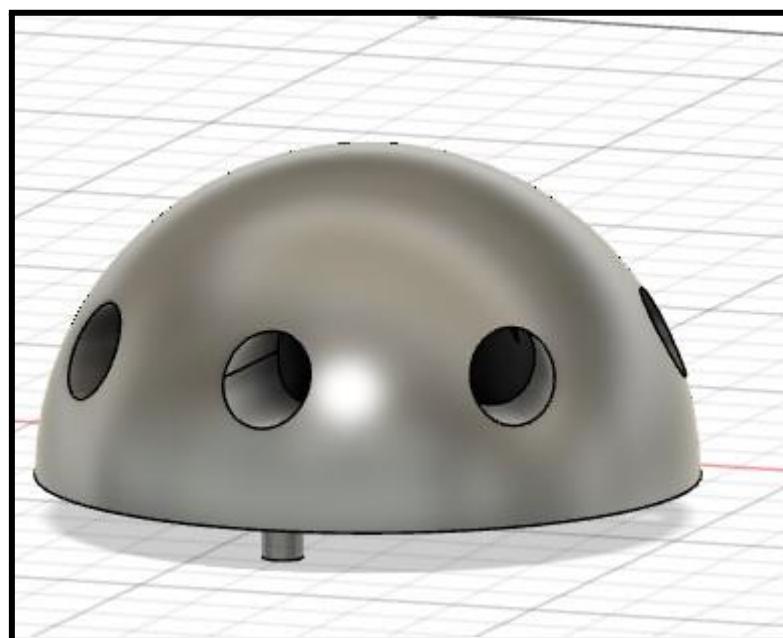
Annexe C.6 : *Assemblage global de la conception de la coque de notre robot (lapin)* :
Équipe IUT Evry



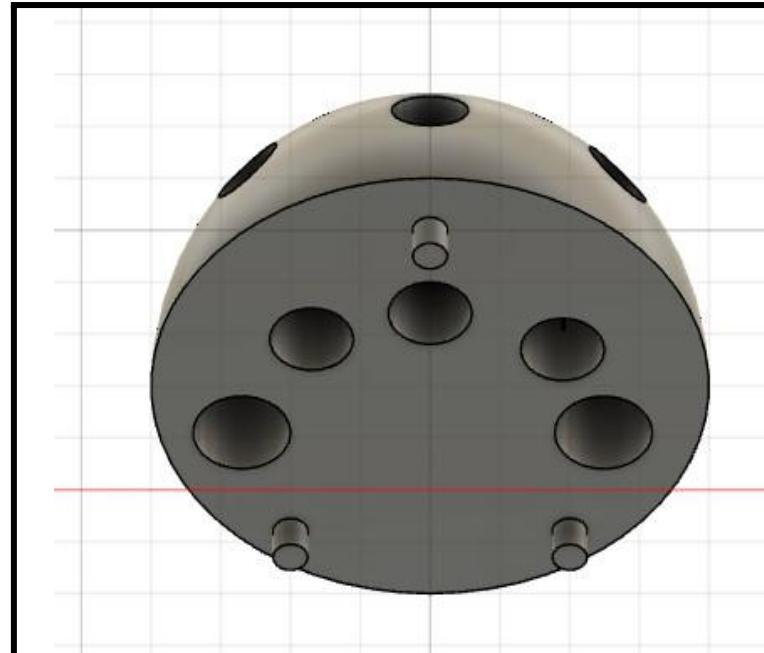
Annexe C.7 : *Support récepteur infrarouge (vue du dessus) (lapin)* : Équipe IUT Evry



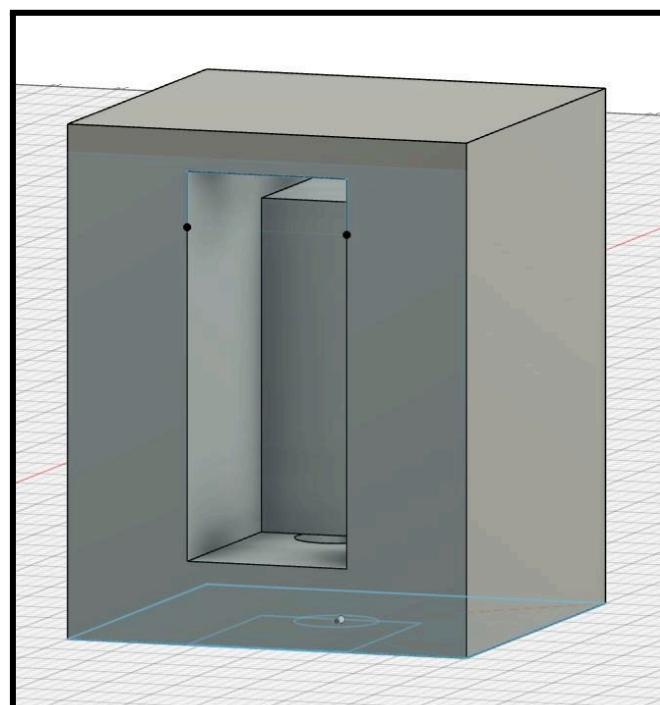
Annexe C.8 : *Support récepteur infrarouge (vue du dessous) (lapin)* : Équipe IUT Evry



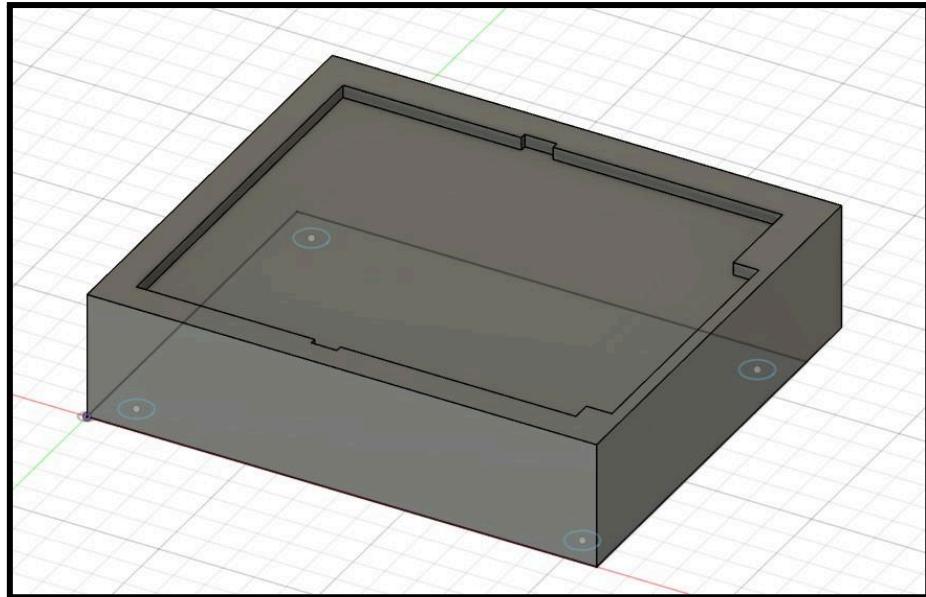
Annexe C.9 : *Cache récepteur infrarouge (vue du dessus) (lapin)* : Équipe IUT Evry



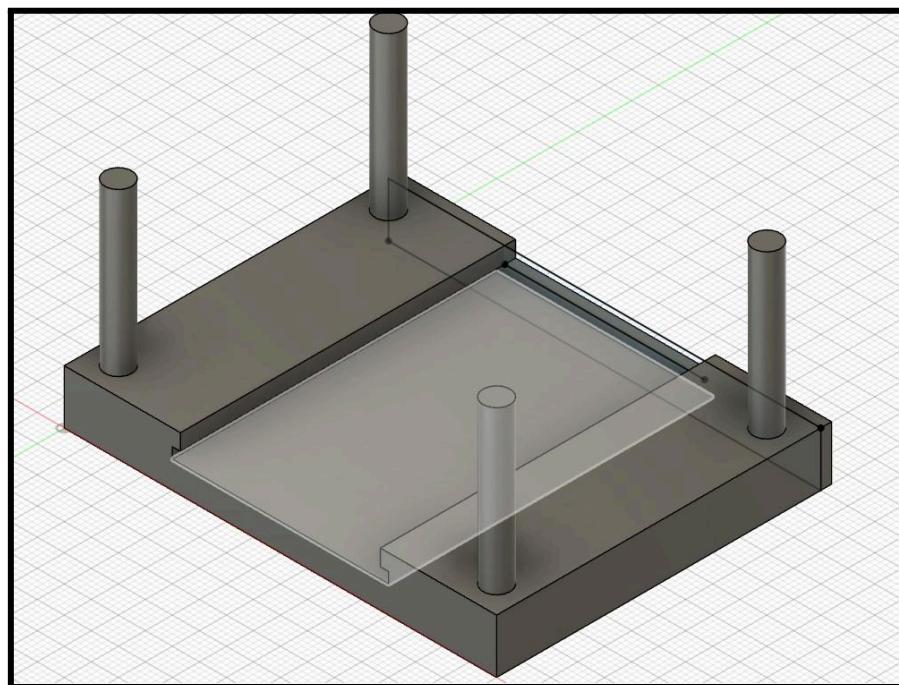
Annexe C.10 : Cache récepteur infrarouge (vue du dessus) (lapin) : Équipe IUT Evry



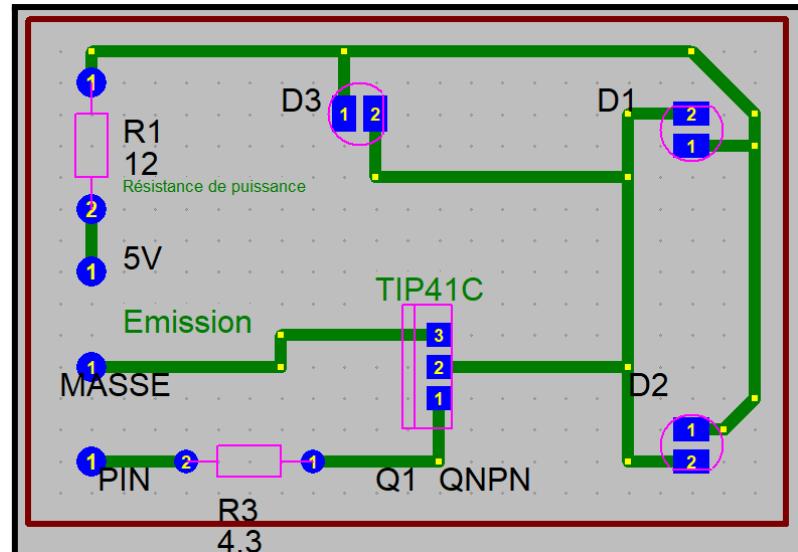
Annexe C.11 : Support servomoteur (lapin) : Équipe IUT Evry



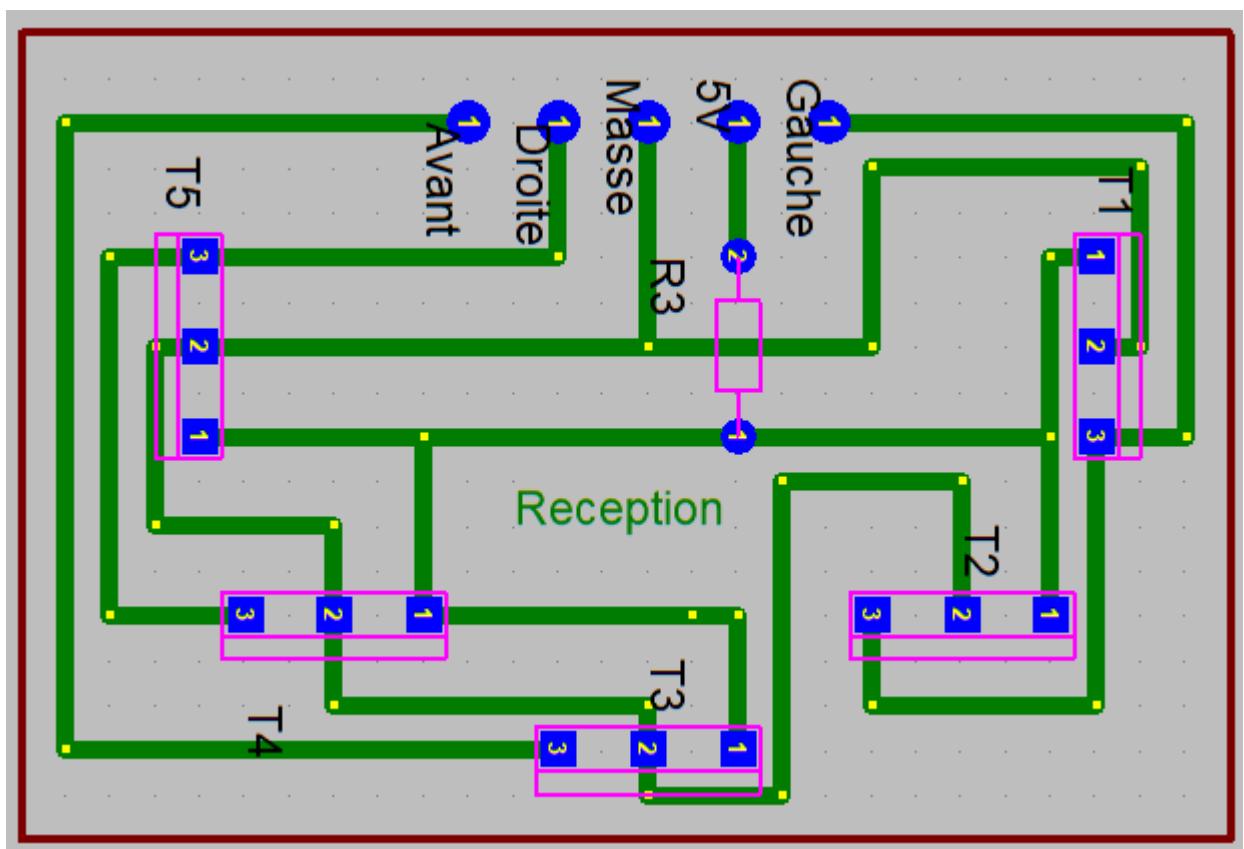
Annexe C.12 : *Support carte arduino (vue du dessus) (balise)* : Équipe IUT Evry



Annexe C.13 : *Support balise (vue du dessus) (balise)* : Équipe IUT Evry



Annexe D.1 : Typon balise émission : Équipe IUT Evry



Annexe D.2 : Typon réception infrarouge : Équipe IUT Evry