



# Suicide Quad

## Projet S4

### Première Soutenance

Vianney Marticou, Florian Ruiz,  
Baptiste Cormorant, David Tchekachev

Professeurs référents : Khelifa Saber et Loïc Blet

2023

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Expériences Personnelles</b>	<b>2</b>
2.1	Vianney . . . . .	2
2.2	Baptiste . . . . .	2
2.3	Florian . . . . .	3
2.4	David . . . . .	3
<b>3</b>	<b>Origine et nature du projet</b>	<b>3</b>
<b>4</b>	<b>Objet d'étude</b>	<b>4</b>
4.1	Synopsis . . . . .	4
4.2	Pré-requis . . . . .	4
<b>5</b>	<b>Répartition des taches</b>	<b>6</b>
<b>6</b>	<b>Etat de l'avancement</b>	<b>6</b>
6.1	Achat . . . . .	6
6.2	Bill Of Material . . . . .	7
6.3	Circuit Imprimé . . . . .	7
6.3.1	Alimentation . . . . .	8
6.3.2	Capteurs . . . . .	9
6.3.3	Alimentation des Moteurs . . . . .	9
6.3.4	Moteur . . . . .	10
6.3.5	Debug . . . . .	10
6.4	Mécanique . . . . .	11
6.5	Algorithmie . . . . .	12
6.5.1	Filtre de Kalman . . . . .	12
6.5.2	Régulateur PID . . . . .	13
6.6	Communication . . . . .	17
6.6.1	API serveur . . . . .	17
6.6.2	Communication coté client . . . . .	17
<b>7</b>	<b>Remerciements</b>	<b>19</b>
<b>8</b>	<b>Conclusion</b>	<b>19</b>

# 1 Introduction

Au sein de ce cahier des charges, nous détaillerons :

- l'origine et la nature du projet : D'où vient l'idée de ce projet ? Quel est son origine ? De quelle nature est-il ?
- l'objet d'étude : Quels sont les buts et intérêts de ce projet ? Qu'est-ce que cela peut nous apporter à l'échelle de groupe et à l'échelle individuelle ?
- l'état de l'art : Quel est la première application de ce type ? Quels sont les principales autres outils de ce type existants ? Quels sont leurs points forts ? Quelles sont leurs fonctionnalités propres ?
- le découpage du projet : Comment sera répartie la charge de travail au sein des membres du groupe ? Quels sont les principales phases de développement ?
- le planning : Quels sont les principales dates importantes du projet ? Dates de soutenance, date de sortie des fonctionnalités, etc.
- la répartition des tâches : Qui sera en charge de quels fonctionnalités du projet ? Qui sera en charge de gérer la communication Client-Serveur ? Le site Internet ?

## 2 Expériences Personnelles

### 2.1 Vianney

Pendant mon temps libre, j'ai eu l'occasion de programmer des ATmega328p en bare-metal avec et sans Arduino IDE. Je suis entrain de découvrir le monde des STM32, les micro-controllers que nous allons utiliser dans ce projet. Je me passionne aussi pour la modélisation 3D, cela m'a permis de créer de nombreux assemblages. J'ai aussi pour avantage d'être une personne extrêmement curieux et motivé.

### 2.2 Baptiste

J'ai travaillé sur des projets de création de sites web et d'applications. J'ai ensuite découvert l'univers de l'électronique avec l'utilisation d'Arduino et de Raspberry Pi, que j'ai utilisés dans le cadre de mini-projet. Grâce à ces expériences, j'ai développé une bonne compréhension de la programmation, de l'électronique et de la robotique, et j'ai pu mettre ces compétences en pratique dans divers projets personnels.

## 2.3 Florian

J'ai pu dans le passé utilisé arduino IDE, et de participer à la création d'un circuit imprimé, dans l'association Evolutek afin de créer une vitrine pour la coupe de robotique de l'année dernière. J'ai aussi suivie les cours d'arduino de l'année dernière en NTS. Et pour finir je suis en ce moment sur un projet qui vise a mettre une écran LCD et un haut-parleur sur une carte arduino UNO, pour un cadeau. Je pense donc beaucoup m'amuser durant ce projet et de découvrir en profondeur le domaine des robots.

## 2.4 David

En classe de Terminale, pour les olympiades de SI, j'avais déjà eu à travailler sur un projet plutôt similaire. Avec mon groupe nous avons assemblé une voiture contrôlée par un micro-ordinateur (Raspberry Pi) et équipée de multiples capteurs, dont des capteurs de distance à base d'ultra-sons, une caméra Pixy pour la reconnaissance du chemin à suivre, d'un module Bluetooth/WiFi pour la communication avec un téléphone ou ordinateur. J'ai donc une petite expérience sur ce genre de projet et j'espère pouvoir l'apporter à ce projet pour le mener à bien.

# 3 Origine et nature du projet

Notre projet vise à combiner les aspects de la robotique matérielle et de l'informatique en mettant en œuvre des techniques de robotique mobile et de SLAM pour créer un robot capable de cartographier un labyrinthe et de trouver le plus court chemin entre le point de départ et d'arrivée. Il s'agit d'un projet à la fois technique et pratique qui vise à développer les compétences en matière de robotique matérielle, d'électronique et de programmation en C.

En plus de la réalisation d'un robot capable de cartographier un labyrinthe, nous voulons également mettre en place un système de communication pour envoyer les données récoltées par le robot à un serveur distant. Ce serveur sera utilisé pour reconstituer la carte effective du labyrinthe et pour trouver le plus court chemin entre le point de départ et d'arrivée. Cet aspect du projet est particulièrement intéressant car il nous permettra de mettre en pratique les concepts de réseau en C, en utilisant des techniques de communication client-serveur et de lecture de paquets.

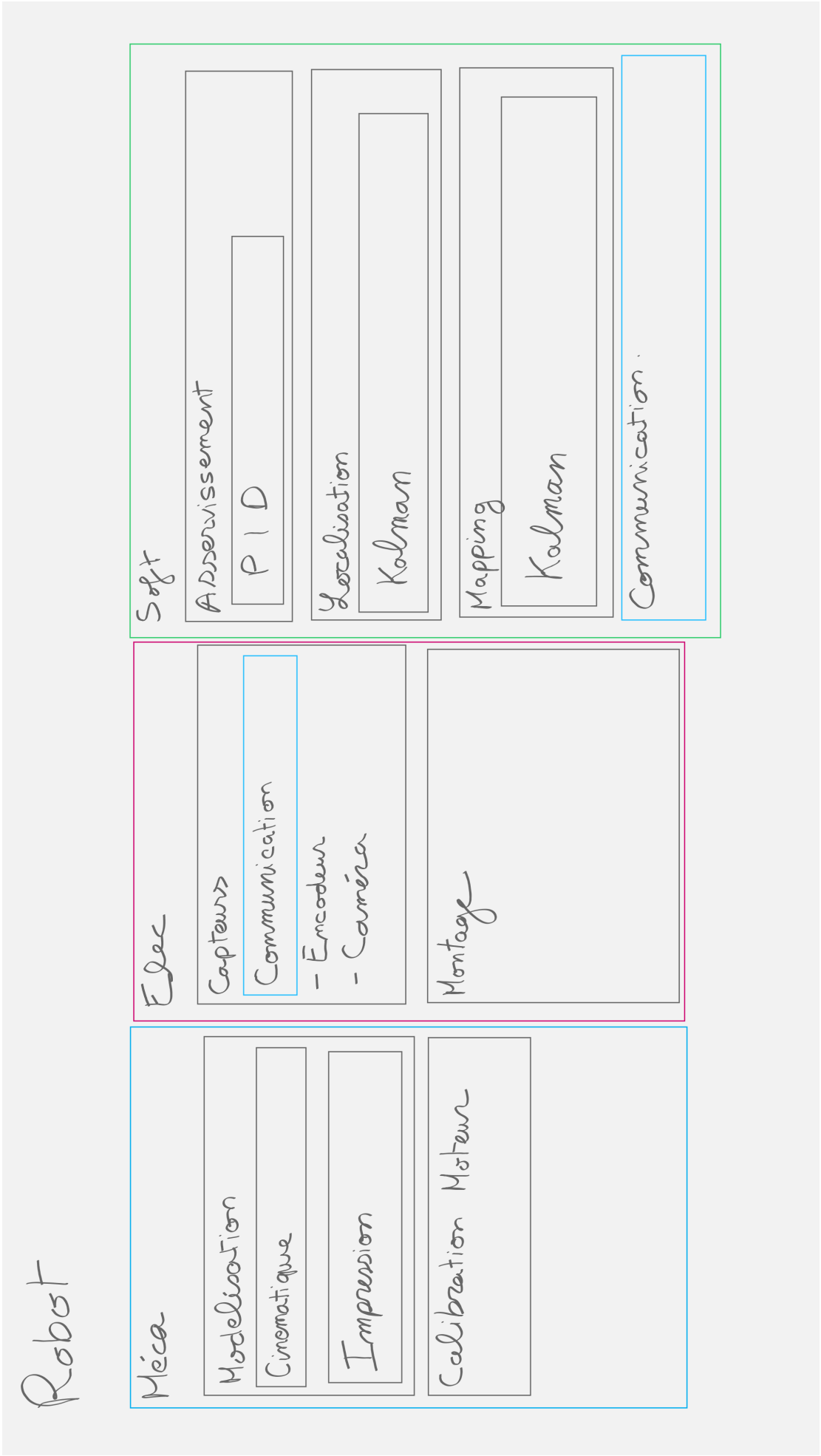
## 4 Objet d'étude

### 4.1 Synopsis

Le but de notre projet va être de créer de toutes pièces un robot qui aura pour mission de cartographier un labyrinthe. En parallèle de la cartographie, il devra envoyer l'intégralité des données récoltées à un serveur distant. Ce serveur aura pour tâche de reconstituer la carte effective du labyrinthe puis de trouver le plus court chemin entre le point de départ et d'arrivée.

### 4.2 Pré-requis

- Le robot : Le robot devra pouvoir se déplacer sur un terrain plan. Le labyrinthe étant représenté par une feuille, on pourra considérer qu'il n'y aura pas de pente ou d'obstacle sur son chemin. Ce robot devra aussi pouvoir "voir" la feuille et son environnement. Afin de reconnaître, les murs (traits noirs) et les points de départ et d'arrivée. Une autre de ses compétences sera de pouvoir échanger avec le monde extérieur, pour envoyer les données récoltées sur le serveur.
- Le serveur : Il devra être capable de gérer la récolte de données via le robot et de reconnaître que le robot a fini son travail. Une fois, le travail du robot fini, il devra reconstituer de manière visuelle le labyrinthe et rendre compte à l'utilisateur du chemin le plus court entre les deux points.
- Le labyrinthe :  
Il sera symbolisé par une feuille blanche sur laquelle, on aura préalablement dessiné des murs (traits noirs).
- Les points :  
Les points de départ seront symbolisés par des ronds de couleur dans le labyrinthe. Le robot devra donc interpréter les formes. Il devra pouvoir faire la différence entre une ligne (un mur) et un rond (un point). Une fois cette distinction faite, il faudra aussi gérer les différentes couleurs entre le point de départ et d'arrivée.



## 5 Répartition des tâches

Pour rappel, voici la répartition de tâches de notre projet tel que nous l'avons mis sur notre Cahier des Charges.

<i>Tâches</i>	David	Vianney	Florian	Baptiste	Soutenance
—- Mécanique —-					
Cinématique		X			1-2
Impression	X	X	X	X	1-2
Calibration Moteur		X			1-2
—- Electronique —-					
Encodeur Rotatif		X			1-2
Caméra	X				2
Montage	X	X	X	X	1-2
Communication				X	2-3
—- Logiciel —-					
Filtre Kalman	X	X			2-3
Localisation	X	X	X	X	2-3
Mapping	X	X	X	X	3
Régulateur PID		X	X		1-2
—- Serveur —-					
Communication	X			X	2-3
GUI	X				3
Recherche du plus cours chemin		X		X	3

## 6 Etat de l'avancement

### 6.1 Achat

Un des principaux facteurs à prendre en compte est l'achat des composants. Sans les composants, le projet ne peut pas réellement avancer. Nous avons pu commander la plupart des composants dans un temps raisonnable. Cependant nous ne les avons pas tous réceptionner. On pourra notamment citer la carte Wifi et la caméra dans la liste des éléments manquants. Heureusement, ce ne sont que des composants secondaires de notre projet, du moins dans un premier temps.

## 6.2 Bill Of Material

Dans cette partie, nous allons faire un rapide point sur le cout de ce projet.

Composants	Prix	Quantité	Utilisation
Rotary Encoder	23,46 €	2	Lit la distance parcourue
Nucleo-F446ZE	16,47 €	1	Le cerveau de notre projet
Centrale inertielle	7,11 €	1	Localisation du robot
Resistance ( $10\Omega$ , $10k\Omega$ , $12k\Omega$ )	0,84 €	24	Élément résistif
Pont en H	12,23 €	4	Gestion des moteurs
Traco 24V - 5V	5,63 €	1	Convertisseur de Tension
Traco 24V - 1.5V	4,69 €	1	Convertisseur de Tension
Traco 24V - 3.3V	4,69 €	1	Convertisseur de Tension
Traco 24V - 12V	4,69 €	1	Convertisseur de Tension
Condensateur 100 $\mu F$	0,16 €	9	Lissage des variations
JST (toutes tailles confondues)	3,57 €	35	Connecte les différents câbles
Circuit Imprimé	8,60 €	1	Circuit Imprimé
Boutton (toutes tailles confondues)	1,25 €	4	Aide au Debug
XT-60 (toutes tailles confondues)	2,54 €	7	Connecte les différents câbles
Support Fusible	1,65 €	2	Sécurise
Dupont F - M & M - M	4,58 €	2	Supporte la Nucleo sur PCB
Condensateur 0.1 $\mu F$	0,14 €	7	Lissage des variations
Total	102,3 €		

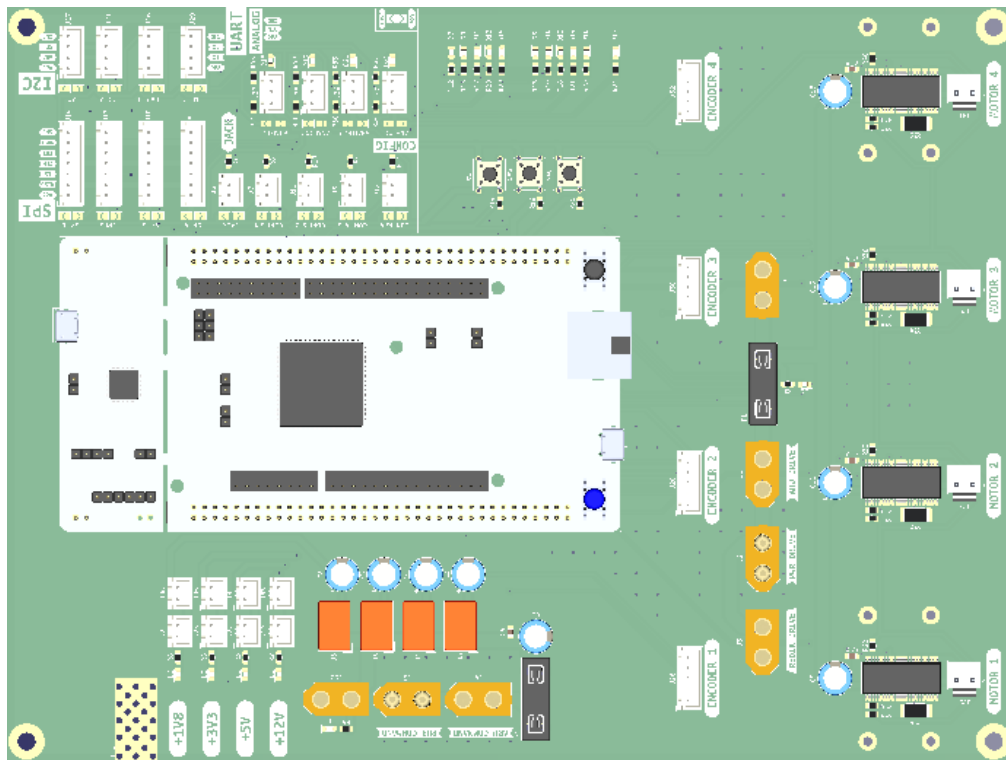
## 6.3 Circuit Imprimé

Afin que ce projet puisse se finir, nous avons acheter un Circuit Imprimé. Cela va nous permettre éliminer un certain nombre de câbles, notamment de faux contact. De plus, on enlever le fameux problème du câble qui est débrancher et d'on personne ne sait où il va.

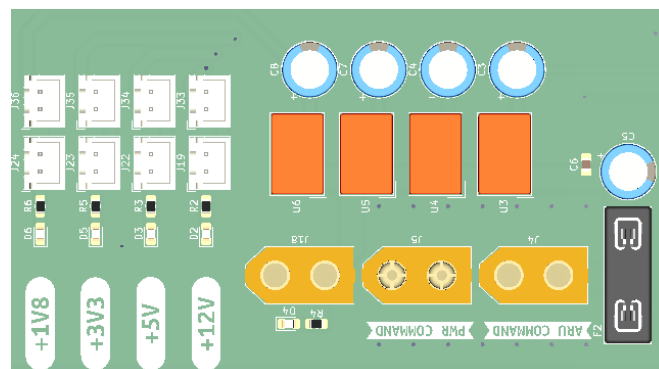
Un des intérêts d'avoir un circuit imprimé est celui de la professionnalisation. Effectivement, je ne pense pas qu'il y a de débat sur le fait qu'un circuit imprimé fait bien plus professionnel qu'un ramassis de câble.

L'avantage que nous apporte un circuit imprimé est l'accès a des composants électroniques de taille bien plus faible. Par exemple, les condensateurs de 100 micro Farad, que l'on appelle les composants 0805, sont des éléments de 1,25mm de largeur. Il faut avouer que souder ce genre de composants n'est pas vraiment simple. Mais cela, nous permet de ne pas avoir avoir à nous soucier de la taille de notre robot.





### 6.3.1 Alimentation

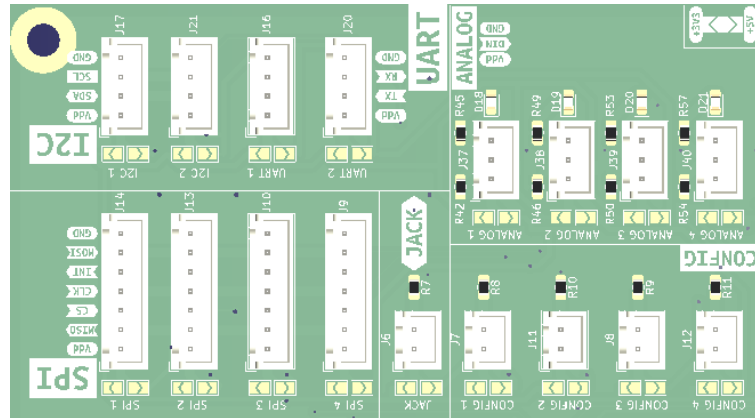


Cette partie de la carte permet de faire toute la gestion de l'alimentation du robot. Les blocs représentés en oranges sur la modélisation sont des traco. Ils permettent de convertir une source de tension, en élevant ou abaissant sa tension. Dans notre cas, nous avons équipé notre robot :

- Un traco 24V vers 12V
- Un traco 24V vers 5V
- Un traco 24V 3.3V
- Nous avons laissé libre l'emplacement du convertisseur 12V vers 1.8V. Pour le moment, aucun de nos capteurs est alimenté en 1.8V. Néanmoins nous ne nous fermons aucune porte surtout s'agissant des capteurs.

De plus, nous avons souder les connecteurs blancs, des JST, ils nous seront utile lorsque nous devrons alimenter nos capteurs.

### 6.3.2 Capteurs



Cette partie de la carte est la zone, la plus "futur proof". En informatique, il existe un grand nombre de composants qui permettent de faire un infinité de choses. Cependant le revers de la médailles est que pour une multitude de possibilité, une multitude de moyen de communication. Heureusement pour nous, il n'existe qu'une poignée de moyen de communication (du moins répandue). Nous appelons donc les passer en vue et voir une des utilisations possible de ces protocoles.

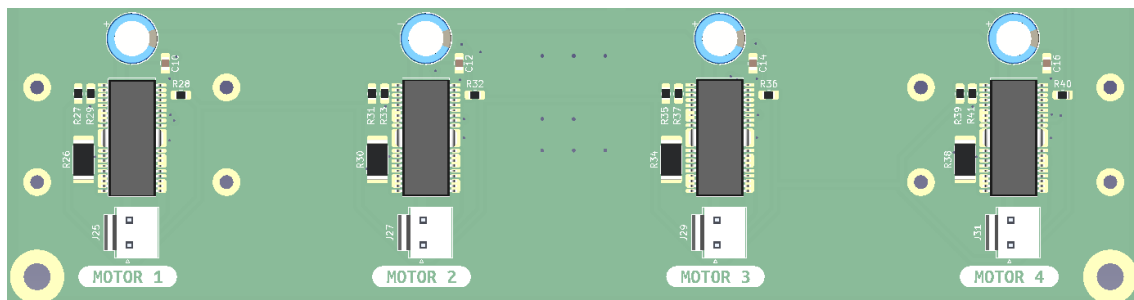
- **SPI** (Serial Peripheral Interface) : C'est un mode communication qui repose sur le principe de maître - esclave. Le maître contrôle la communication. Ce protocole a l'avantage de synchronisé les deux composants sur l'horloge et d'avoir un meilleur débit de l'UART et que le I2C. Cependant il monopolise plus de Pin que ces derniers.
- **UART** (Universal Asynchronous Receiver Transmitter) : Comme son nom l'indique, c'est un protocole qui repose sur une communication asynchrone entre un émetteur - receveur. L'UART repose sur des vitesses de transmission particulière, par ex : 9600 baud.
- **I2C** (Inter-Integrated Circuit) : Ce protocole reprend le concept de maître - esclave du bus SPI, mais il rajoute le fait qu'un esclave peut prendre la position de maître, en inversant les rôles.
- **JACK** : C'est un canal qui achemine des signaux logiques.
- **ANALOGIQUE** : De même, c'est un canal qui achemine des signaux analogiques.

### 6.3.3 Alimentation des Moteurs



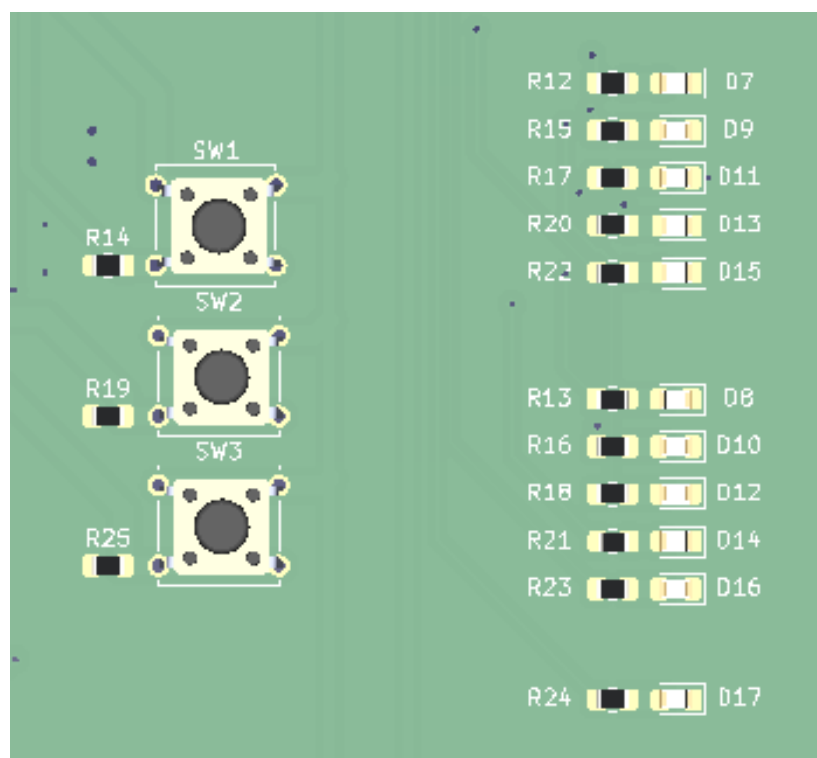
Dans cette partie, il y a 4 connecteurs JST qui permettent de récupérer les données des encodeurs rotatifs. Ils nous seront très utile pour faire le Régulateur PID. De plus, il y a les connecteurs XT-60 qui ont pour mission de gérer l'alimentation des moteurs. Nous avons aussi un connecteur sur lequel nous connectons un arrêt d'urgence. En effet, nous n'avons pas encore parlé de la partie sécurité de notre projet mais nous devons absolument pouvoir couper l'alimentation faite aux moteurs.

### 6.3.4 Moteur



Cette zone contient la gestion des différents moteurs a chaque moteur est associé un composant que l'on appelle un driver. Il permet de contrôler le moteur, une de ses principales taches est d'inverser la bornes - et la borne +. En effet, le moteur est un dipôle "réversible". Les guillemets sont nécessaires car un moteur n'est pas a proprement dit un dipôle réversible dans le sens, ou si l'on inverse les bornes le moteur marchera mais dans le sens inverse.

### 6.3.5 Debug



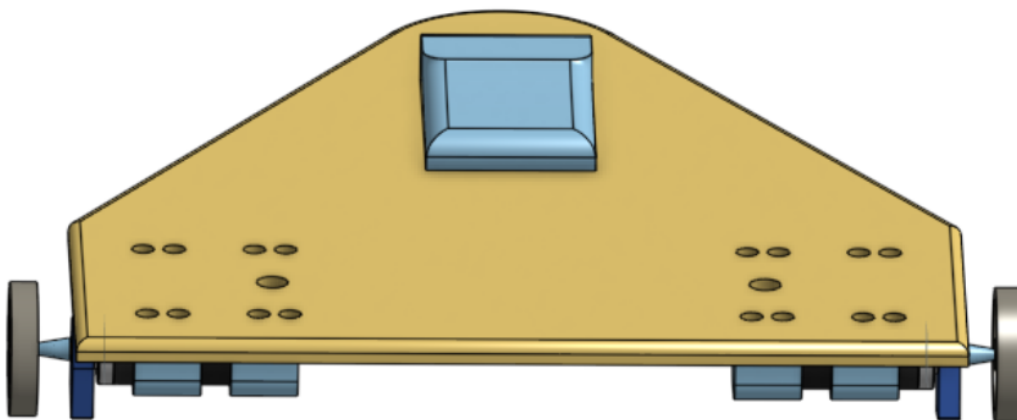
Ce morceau a pour seul et unique but de faire du debug sur la carte. Nous avons trois boutons que l'on link avec des actions particulière. Ainsi que 11 Led de couleur différentes qui nous permettent d'afficher des informations. Par exemple, un niveau de criticité, si les receveur des protocoles sont connecter ou bien juste montrer que le robot est allumé.

## 6.4 Mécanique

Afin de voir la viabilité de notre robot, nous avons modéliser une grande partie de ses pièces en 3D. Tous cela dans l'optique de pouvoir réaliser un assemblage virtuel de notre projet. On peut dire que nous avons créer un digital twin de notre robot.

Ce domaine nous a donné un peu de fil a retordre, un des principaux était que l'on devions créer une pièce qui devait pouvoir résister aux contraintes suivantes :

- Le poids du robot, c'est une information qu'il ne faut négliger notre robot va contenir de nombreux composants dont des batteries.
- La vitesse angulaire de notre robot, nous avons pris des moteurs qui sont légèrement sur dimensionner pour notre projet. Maintenant nous allons devoir assumer cela.
- La faisabilité de cette pièce, nous sommes nos propres usineurs. Et malheureusement pour nous, nous n'avons pas toutes les machines et connaissances du monde. Pour faire cette pièce, nous avons opter pour l'impression 3D, bien que cette méthode contiennent de nombreux désavantages



## 6.5 Algorithmie

Pour l'instant, nous avons beaucoup parlé de l'aspect physique de notre projet. Maintenant, nous allons aborder le sujet de l'algorithmie de notre robot, qui est le coeur de notre métier et de notre projet.

Il faut savoir que nous n'avons pas encore commencé à coder notre robot proprement dit. Nous avons préféré nous concentrer sur l'aspect électronique et mécanique du robot, afin de ne plus avoir ce problème en tête lors des prochaines soutenances.

### 6.5.1 Filtre de Kalman

#### 6.5.1.1 Explication

Le filtre de Kalman est un filtre dit à réponse impulsionnelle infinie, c'est à dire qu'il se base sur les valeurs reçues des capteurs (impulsionnel), qui sont bruitées ou incorrectes. Il prend aussi en compte toutes les valeurs antérieures (infini) ce qui lui permet d'être de plus en plus précis au fur-et-à-mesure qu'il reçoit des mesures. Dans notre cas, ce filtre nous sera très utile car il nous permettra de corriger dynamiquement le déplacement de notre robot car nous savons par expérience que les capteurs que nous allons utiliser ne sont pas parfaits et auront du bruit dans les mesures qu'ils fourniront au contrôleur.

#### 6.5.1.2 Fonctionnement

Sans trop rentrer dans les détails trop techniques, voici comment fonctionne le filtre de Kalman :

- On définit la position de départ de notre robot, souvent à l'origine de l'espace (0, 0, 0)
- On définit les constantes de bruit qu'on a trouvé durant les phases de tests, qui nous permettent de savoir à quel point la source de donnée n'est pas fiable.
- Ensuite, à chaque itération :
  - On se base sur la dernière estimation ou position connue
  - On calcule le coefficient d'incertitude (aussi appelé gain de Kalman) de notre mesure en fonction de l'incertitude de la mesure précédente, cette valeur tend vers une constante plus on a de mesures.
  - On récupère la dernière mesure du capteur
  - On calcule la nouvelle estimation de position en se basant sur la dernière estimation + la différence entre la dernière mesure et la dernière, elle-même corrigée par le coefficient d'incertitude. Ce qui nous donne cette formule :

$$\text{pos\_est} = \text{last\_pos\_est} + \text{incert\_coeff} * (\text{pos\_meas} - \text{last\_pos\_est})$$

### 6.5.1.3 Procédure de test

Maintenant qu'on a globalement compris comment fonctionne ce filtre et comment il corrige les données reçues des capteurs, il va falloir le faire tourner sur notre robot afin de déterminer les constantes avec précision.

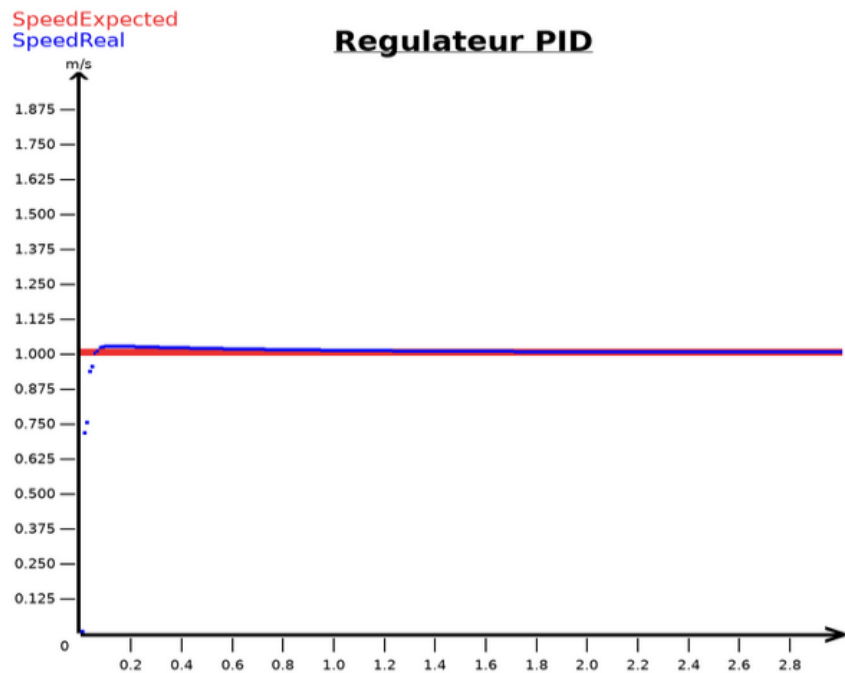
Pour cela, il faudra faire déplacer sur un parcours déjà connu et mesuré afin de voir quelles valeurs les capteurs nous retournerons, et ensuite ajuster les constantes afin d'obtenir un système qui arrive à sa position avec précision grâce aux capteurs embarqués, sans avoir besoin d'une autre source d'externe pour assurer la véracité des données collectées.

## 6.5.2 Régulateur PID

### 6.5.2.1 Explication

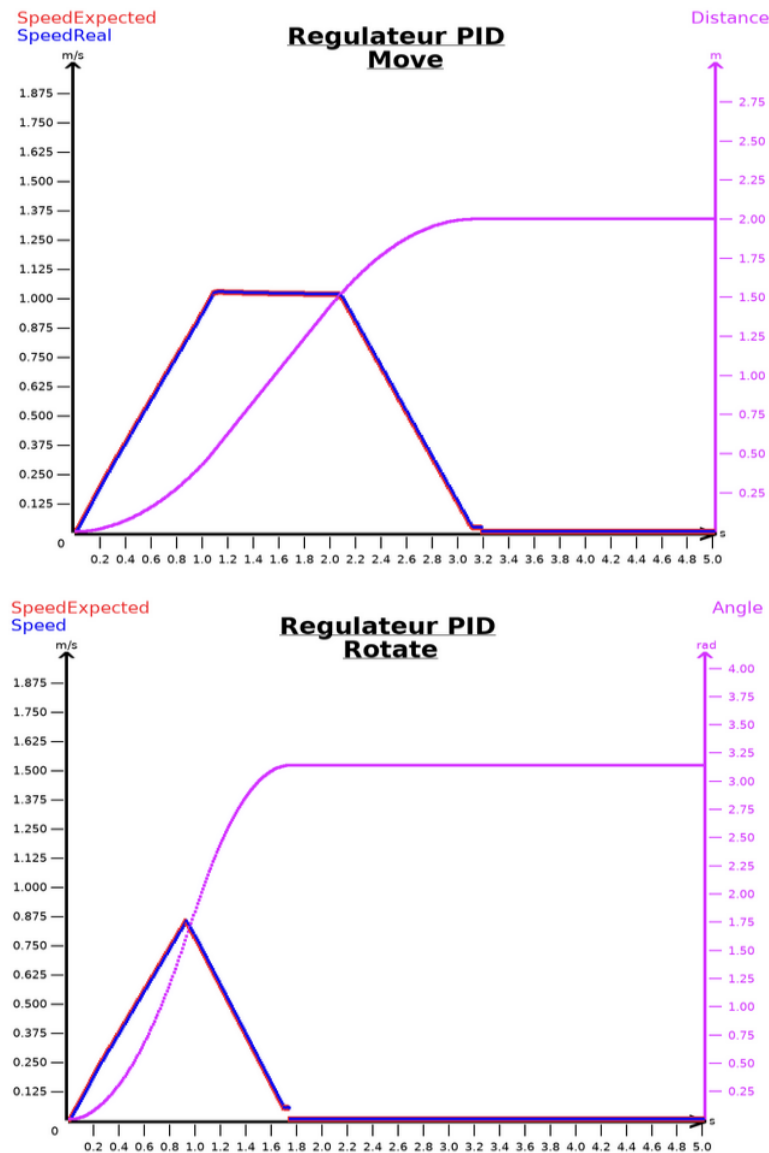
Un régulateur PID est un outil qui permet l'asservissement de notre robot, c'est un élément indispensable à tous robots, en effet il est à la base de ses mouvements et de sa robustesse. Son principe est de réguler la consigne qui est envoyée aux moteurs, enfin qu'ils atteignent un objectif progressivement et qu'ils s'y stabilisent. Il permet aussi que notre robot reste à une position fixe même si il est poussé et d'avancer à une distance donnée même si il est retenu pendant sa trajectoire. Ceci permet en cas de ralentissement inopiné de notre robot qu'il conserve sa trajectoire et que son objectif soit atteint.

Le régulateur PID est donc un outil qui permet cette régulation. Il se repose sur des calculs basés sur la différence entre l'objectif voulu et la valeur réelle mesurée (l'erreur de la consigne) et la valeur réelle mesurée. Le calcul consiste à prendre une valeur Proportionnelle de l'erreur multipliée par un coefficient, d'y ajouter une valeur Intégrale des erreurs (leurs sommes) multipliée par un autre coefficient et d'y ajouter une valeur Dérivée de la valeur mesurée (la différence entre les deux dernières valeurs mesurées) multipliée par un dernier coefficient. D'où le nom de régulateur PID (Proportionnelle Intégral Dérivé). Dans le concept on peut comprendre son principe comme un humain qui conduirait une voiture qui est à l'arrêt et qui veut aller à 100 km/h. Tout d'abord si on ne prend que le terme proportionnel, cela reviendrait à appuyer fort quand on est bas de l'objectif, ainsi une fois arriver à 100 km/h, nous allons relâcher l'accélérateur car on a dépassé les 100 km/h, et une fois redescendu en dessous des 100 km/h on ramène sur l'accélérateur. On va ainsi tendre vers une vitesse inférieure à notre objectif, ici 90 km/h. Pour régler ce problème on rajoute le terme intégral, cela revient maintenant à accélérer plus fort quand on reste longtemps éloigné de l'objectif, ainsi il y aura une oscillation pour atteindre la vitesse voulue. Finalement on rajoute le dernier terme celui de la dérivée, il revient lui à moins accélérer lorsqu'on se rapproche de l'objectif, ainsi on dépasse moins l'objectif et on redescend moins en dessous de l'objectif. Finalement on a donc un régulateur qui va rapidement à notre objectif avec peu d'oscillation.



### 6.5.2.2 Mise en place

On va donc maintenant vouloir que notre robot avance d'une certaine distance et tourne d'un certain sens. Pour se faire il ne suffit pas simplement d'exécuter le calcul du PID, en effet il produit une secousse brusque au démarrage et une fois la distance voulu atteinte, on ne peut pas arrêter d'un coup notre robot à pleine vitesse. On doit lui faire suivre un trapèze de vitesse. C'est à dire qu'il doit, au début, accélérer uniformément puis rester à sa vitesse désiré et par la suite décélérer. Il suffit donc pour cela de diviser le mouvement en trois phase. Au début la vitesse voulu de notre robot commence à la vitesse minimum pour augmenter à intervalle régulier. A partir de là, il y a deux possibilité, soit la vitesse limite est atteint alors la vitesse stagne, on prend en compte la distance qu'il a fallut pour accélérer pour pouvoir décélérer de la même façon, soit on atteint la moitié de la distance voulu avant la fin de l'accélération alors on décélère. Pour la décélération la diminution de la vitesse doit se faire 1.05 fois plus rapidement que l'accélération sinon la distance parcouru est atteinte avant la fois de la décélération. Cela est dû au fait que la vitesse augmente plus rapidement quand on demande une vitesse plus haute que la vitesse actuel du robot. Ensuite après la décélération le robot se stabilise à une vitesse minimum qui lui permet de parcourir les derniers centimètres. Ainsi si on dessine la courbe de la distance parcouru dans le temps, on voit que le départ se fait progressivement, ensuite l'évolution est stable et sur l'arrivée la distance atteint la distance désiré progressivement. Pour tourner on effectue la même démarche sauf que dans la pratique on fait tourner les roues dans le sens opposé et on calcul l'angle parcouru avec la distance parcouru par une roue.



### 6.5.2.3 Suites de test

Pour pouvoir tester notre algorithme, il a d'abord fallu simuler d'utiliser nos roues et dire à notre algorithme qui déplace notre robot de combien avait avancé nos roues. Pour faire simple nous avons dit que nos roues suivaient parfaitement la consigne qu'envoyait le régulateur PID et qu'aucune perturbation déréglaient notre robot. Il a fallu avoir un retour visuelle également. Tous d'abord il s'agissait d'un fichier texte et de la console où à chaque étapes on notait les valeurs enregistré/calculé de l'encodeur, l'erreur par rapport à la vitesse demandé, la distance parcouru par la roue depuis le dernier calcul, le temps qu'il a fallu et la distance totale parcouru. Toutefois ceci est utile pour comprendre l'origine des erreurs de calcul mais pas pour suivre le robot dans le temps. Il a donc fallu modéliser un graphique, à l'aide de SDL2, pour pouvoir suivre la vitesse demandé au robot, la vitesse mesuré sur nos roues et la distance parcouru. On avait donc un moyen de comprendre les erreurs et de visualiser nos valeurs et voir si notre trajectoire était satisfaisante. Pour se faire on dessine un point à chaque intervalles de temps et une fois tous collé, on obtient une courbe et des droites.



Maintenant qu'on observe notre mouvement on peut le régler, en effet pour calculer le PID, on utilise des coefficients devant chaque terme, on peut donc maintenant les choisir. Nous avons décidé de les choisir par tâtonnement, on d'abord réglé celui du P, ensuite celui du I et celui du D par la suite. Mais cela n'est pas simple car nous notre robot ne doit avoir aucune secousse, ces coefficient sont donc très bas et à chaque coefficient rajouté on doit rerégler les anciens.

```
Rep: 142 -> 1.492014s  1.066003m
For Left
>...ValEncodeur = 0.027011
>...Distance = 0.010178
>...Speed = 1.017775
>...Error = -0.007776
>...Commande = 1.017663
For Right
>...ValEncodeur = 0.027011
>...Distance = 0.010178
>...Speed = 1.017775
>...Error = -0.007776
>...Commande = 1.017663
```

```
Rep: 465 -> 4.993821s  -0.000094m
For Left
    ValEncodeur = 0.000000
    Distance = 0.000000
    Speed = 0.000000
    Error = 0.030000
    Commande = 0.000000
For Right
    ValEncodeur = 0.000000
    Distance = 0.000000
    Speed = 0.000000
    Error = 0.030000
    Commande = 0.000000
```

#### 6.5.2.4 Finalité

Notre algorithme est capable d'indiquer à notre robot la vitesse à laquelle il doit aller pour avancer ou tourner. Avec les valeurs que nous avons l'ordre de grandeur d'erreur de parcours est entre le millimètre et le micromètre. Toutefois ceci n'est pas un problème majeur car il connaît son erreur par rapport à la distance ou l'angle voulu, ainsi quand on calculera sa position dans le labyrinthe, il n'y aura pas de problème car il rajoutera l'erreur à la distance désiré.

## 6.6 Communication

### 6.6.1 API serveur

L'API mise en place pour la communication entre le robot de cartographie de labyrinthe et le site Web permet de faciliter l'échange de données entre ces deux entités. Cette API offre des fonctionnalités qui permettent au site Web de suivre l'avancement du robot et d'afficher des graphiques et des schémas pour visualiser la cartographie.

L'API offre également des fonctionnalités de débogage pour aider à résoudre les problèmes éventuels rencontrés par le robot lors de son parcours dans le labyrinthe. Grâce à cette API, le robot envoie les données de cartographie au site Web via les endpoints définis, permettant ainsi au site de les afficher en temps réel.

Le site Web peut également envoyer des commandes au robot pour contrôler son mouvement et son comportement, en utilisant les endpoints appropriés de l'API. Toutefois, pour que ces commandes soient exécutées, le robot doit être programmé pour interpréter et exécuter les commandes reçues.

L'api a été faite en JS, sur l'environnement d'exécution node js à l'aide des framework express.js et REST pas en C pour des questions de pratique et surtout que l'intérêt du projet n'est pas de faire une API mais de faire un robot qui résout un labyrinthe.

En résumé, l'API REST facilite la communication entre le robot de cartographie de labyrinthe et le site Web. Elle permet de suivre l'avancement du robot et d'afficher des graphiques et des schémas pour visualiser la cartographie. Elle offre également des fonctionnalités de débogage pour aider à résoudre les problèmes éventuels rencontrés par le robot lors de son parcours dans le labyrinthe.

Si on a des difficultés à faire des requêtes http depuis le robot, on se réserve le droit de passer par du TCP comme on l'a déjà fait en tp et ce sera plus simple à implémenter. et donc de ne pas passer par l'API.

### 6.6.2 Communication côté client

Le robot de cartographie de labyrinthe peut communiquer avec l'API REST à travers des appels HTTP en envoyant des données sous forme de requêtes. Les données envoyées par le robot peuvent inclure :

- Les données de cartographie : Le robot peut envoyer les données de cartographie du labyrinthe à l'API, telles que la position des murs, des portes et des pièces. L'API peut stocker ces données dans une base de données ou les envoyer au site Web pour être affichées sous forme de graphiques et de schémas.

- Les données de localisation : Le robot peut également envoyer des données de localisation à l'API, telles que sa position actuelle dans le labyrinthe. Ces données peuvent être utilisées pour suivre l'avancement du robot et afficher sa progression en temps réel sur le site Web.
- Les données de capteur : Le robot peut envoyer les données de capteur à l'API, telles que les informations sur les obstacles et les distances parcourues. Ces données peuvent être utilisées pour évaluer les performances du robot et détecter tout problème éventuel.
- Les données d'état : Le robot peut également envoyer les données d'état à l'API, telles que les erreurs détectées. Ces données peuvent être utilisées pour surveiller l'état du robot et détecter les problèmes qui nécessitent une intervention humaine.

En résumé, le robot de cartographie de labyrinthe peut envoyer des données à l'API REST sous forme de requêtes HTTP. Ces données peuvent inclure les données de cartographie, les données de localisation, les données de capteur et les données d'état. L'API peut stocker ces données dans une base de données ou les envoyer au site Web pour être affichées et analysées.

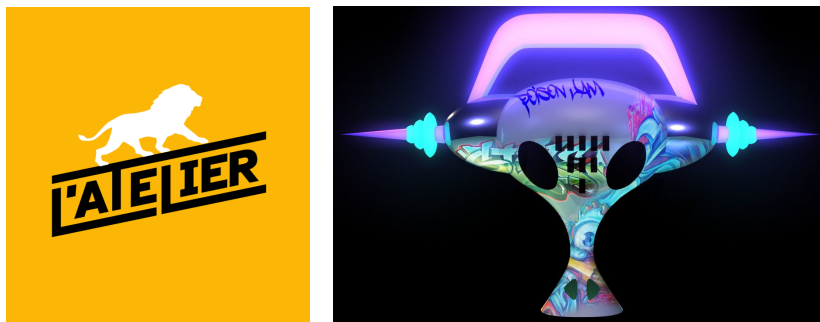
Dans le cas où ne réussirions pas à implémenter en C, les requêtes REST faites à l'API, nous utiliserons un simple transfert de bits brute. Nous devons donc modifier notre serveur pour pouvoir parser les packets puis visualiser les données.

## 7 Remerciements

Nous souhaitons remercier l'association Atelier Lyon qui va nous prêter son matériel et ses locaux. Cela nous permet d'avoir un lieu pour le stockage des composants, du robot mais aussi pour pouvoir tester en condition réelle notre projet.

L'Atelier Lyon est aussi notre mécène, elle va en effet financer la plus grande partie du robot. Sans ce financement, il y a peu de chances que ce projet ait vu le jour. Il faut reconnaître que son prix (env. 300 €) a de quoi décourager les plus téméraires.

Nous remercions aussi Florian Reimat qui nous a fourni le circuit imprimé du robot, sans quoi il aurait été beaucoup plus compliqué de faire ce projet. De plus, il nous a été d'un grand conseil dans tous les aspects pratiques de ce projet, mécanique et électronique.



## 8 Conclusion

En résumé, nous avons bien démarré notre projet mais il nous reste de nombreuses choses à implémenter et expérimenter. Nous avons reçu le circuit imprimé et pratiquement soudé l'intégralité des composants.

Néanmoins, il nous reste encore un grand nombre de choses à terminer et certains composants à acheter. Heureusement, nous allons bientôt entrer dans la partie algorithmique du projet, qui est notre but initial.

Le fait d'avoir quasiment fini la partie matériel du robot, va entre autre motiver le groupe. Nous sommes bien plus confiant maintenant. Il ne faut pas pour autant sous-estimer le travail qu'il reste à achever. Aussi, nous avons bien entamé toute la partie algorithmique avec le filtre de Kalman et le PID, ce qui nous permettra d'aller plus vite une fois que le micro-contrôleur sera connecté car il nous suffira de tout intégrer.