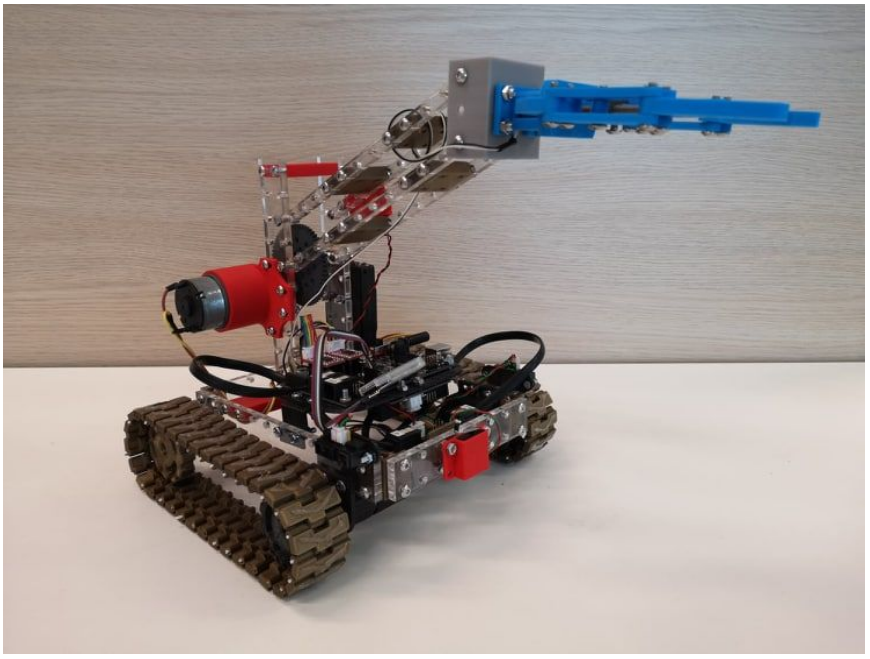
Rapport Projet

P46 : Kit Robot



BERNARD Gaëlle
PITRE Valentin
IMA4 année 2018-2019

Enseignants référents : REDON Xavier, BOE Alexandre

Remerciements

Dans un premier temps nous aimerions remercier l'équipe pédagogique de la spécialité Informatique, Microélectronique et Automatique pour les enseignements qui nous ont permis de réaliser ce projet.

Ensuite, nous voudrions remercier M. Redon et M. Boé pour le temps qu'ils nous ont accordé, pour leurs conseils, pour le matériel prêté et pour les commandes de matériel et de cartes électroniques.

Nous tenons à remercier également Thierry Flamen pour son expertise, son aide et ses conseils lors de la soudure de nos cartes électroniques .

Enfin, nous remercions les Fab-Managers pour leur disponibilité et pour leur aide lorsque nous en avions besoin.

Sommaire

Introduction.....1

1. Analyse et Préparation du Projet.....2
1.1 Pourquoi ce projet ?.....2
1.2 Objectifs.....2
1.3 Positionnement par rapport à l'existant et concurrents.....3
1.4 Scénario d'usage.....4
1.5 Cahier des charges.....5
1.6 Liste des tâches à effectuer.....6

2. Réalisation du Projet.....6
2.1 Choix technique : matériel et logiciel.....6
2.2 Partie Électronique.....7
2.2.1 La carte principale.....7
2.2.2 Les cartes capteurs.....10
2.2.3 Les cartes contrôleur moteur.....12
2.3 Partie Programmation.....13
2.3.1 ATtiny85.....13
2.3.2 Les librairies.....13
2.3.3 La programmation par blocs (Blockly Arduino).....15
2.3.4 L'application Bluetooth.....15
2.4 Partie Modélisation et Mécanique : Les pièces du Kit.....16
2.4.1 Les pièces 2D.....16
2.4.2 Les pièces 3D.....16
2.4.3 Le robot 18

3. Présentation des résultats et remarques 19
3.1 Problèmes rencontrés 19
3.2 Résultat final 20

Conclusion 22
Annexes 23
Annexe 1 : Carte Principale 23
Annexe 2 : Carte Capteurs.....24
Annexe 3 : Caractérisation Capteur Sharp 26
Annexe 4 : Carte Contrôleurs Moteurs 27
Annexe 5 : Programmation 28
Annexe 6 : Catalogue des pièces du Kit.....30
Annexe 7 : Evolution du Robot.....34

Introduction

Les Kits de Robotique sont de plus en plus répandus de nos jours. Ce sont des Kits d'assemblage de robots programmables. Ils contiennent différentes pièces, des moteurs, des capteurs et permettent aux utilisateurs de créer leurs propres robots. Il existe des Kits pour tous les niveaux, du débutant à l'expert en robotique ; les robots sont programmables via une application mobile en Bluetooth ou via des langages de programmations (Scratch, C++, Python ...)

L'inconvénient de ces kits est qu'ils sont souvent assez chers et surtout propriétaires. C'est à dire qu'ils ne sont pas Open Source et que la perspective d'évolution du Kit est limitée à la même marque.

Durant notre 4eme année en spécialité IMA à Polytech Lille, nous avons eu l'occasion de travailler sur un projet de création de Kit de Robotique. L'objectif de ce projet est de proposer une alternative aux Kits de Robotique déjà existants dans le commerce. Pour cela nous allons concevoir un Kit Open-Source (open source en logiciel comme en matériel) permettant l'assemblage d'un robot et une programmation aisée et facilement modifiable.

Dans ce rapport nous allons premièrement vous faire part de notre analyse en amont du projet qui nous a permis de définir les différentes caractéristiques de notre Kit et de déterminer ses étapes de réalisation. Ensuite nous expliquerons chaque étape de réalisation du projet. Puis nous présenterons les résultats obtenus. Enfin, nous conclurons sur ce projet, nous donnerons nos remarques et nos perspectives d'amélioration.

1. Analyse et Préparation du Projet

1.1 Pourquoi ce projet ?

Nous avons étudié certains Kits de Robotique comme les Kits MindStorm NXT, Bioloid et MakeBlock durant nos études. En effet, nous avons utilisé ces Kits lors de nos séances de cours, au Club de Robotique “Robotech” de Polytech Lille ou également lors des Olympiades des Métiers.

Nous avons pu alors nous apercevoir que tous les Kits de Robotique disponibles de nos jours possèdent des points positifs et des points négatifs (autonomie, prix, facilité de programmation, facilité d'assemblage, diversité des modèles disponibles, possibilité d'additionner plusieurs kit...).

En tant que membres de Robotech, nous avons eu à coeur de réaliser ce projet. Il permettra entre autre à Robotech de pouvoir posséder un Kit de Robotique répondant au besoin du club. En effet, Robotech participe chaque année à différents événements (Coupe de France de Robotique, Forum Robotic, MakerFaire…). Ce Kit constituera une base solide pour la fabrication et la programmation des deux robots (primaire et secondaire) de la Coupe de France. Le Kit sera conçu par des étudiants, pour des étudiants.

1.2 Objectifs

Notre Kit sera, à l'inverse de la plupart des Kits existants, Open Source en matériel et en logiciel.

Les utilisateurs pourront eux-mêmes imprimer les pièces du Kit grâce à des fichiers stl (imprimante 3D) et svg (Découpeuse Laser). Ils pourront également reproduire la carte et les différents capteurs, actionneurs, modules de communication grâce aux fichiers partagés et à la liste des composants nécessaires. Cela permettra une économie d'argent considérable et également à l'utilisateur de comprendre en détails avec quels composants sont réalisés tous les éléments du Kit et comment ceux-ci fonctionnent. De plus si un élément du Kit ne fonctionne plus ou est défectueux, il suffit de le réimprimer. Ce n'est pas le cas des Kits déjà existants qui ne sont pas Open Source.

Notre Kit sera simple d'utilisation pour faciliter les branchements et la programmation.

Il se composera d’une carte principale qui permettra de contrôler les capteurs, les actionneurs, les moteurs et les modules de communication. Il possèdera également des cartes pour chaque capteurs proposés et des cartes contrôleur moteur.

Toutes les cartes capteur seront reliés entre eux via RJ25 (protocole I2C). Nous pouvons alors chaîner jusqu’à 128 cartes capteur! Tout cela permettra d’obtenir un câblage complet et facile sans avoir trop de fils.

Le Kit possèdera un système d’alimentation du robot, proposant une autonomie correcte (par exemple batterie lithium). Nous ajouterons également un système de recharge simple et un régulateur de tension adapté aux besoins des composants du robot.

Nous souhaitons également contrôler notre Robot via une application Bluetooth. Nous allons donc utiliser un module Bluetooth pour le Kit.

Ensuite nous réaliserons, à l'aide de la Découpeuse Laser et des imprimantes 3D du Fabricarium de l'école, les différentes pièces du Kit qui peuvent être facilement assemblées pour former plusieurs modèles de robots.

Enfin nous réaliserons différentes librairies proposant un outil de programmation et d'utilisation simplifié du modèle et de ses modules.

Le Kit créé pourra être utilisé au club Robotech de Polytech Lille. Il permettra aux nouveaux membres de se familiariser avec la robotique, d'utiliser les machines du Fabricarium et de créer un robot.

1.3 Positionnement par rapport à l'existant et concurrents

Nous avons étudié nos principaux concurrents afin d'évaluer leurs avantages et leurs faiblesses. Nous pourrons alors définir au mieux les caractéristiques de notre Kit et son cahier des charges afin qu'il soit le plus optimisé possible.

Notre premier concurrent est la marque Makeblock. Cette marque propose différents Kits pour construire des robots.
Voici deux de leurs kits :

Makeblock (mBot V1.1)

Makeblock (Ultimate 2.0)

<u>Avantages des 2 Kits:</u>

- Pièces très résistantes et assemblage robuste grâce aux vis
- Compatible avec plusieurs moyens et langages de programmation
- Compatibles avec des capteurs de marque autre que Mblock
- Facile d'utilisation
- Prix abordable pour le Robot mBot V1.1

<u>Inconvénients des 2 Kits:</u>

- Autonomie limitée
- Prix assez élevé pour l'Ultimate 2.0
- Nombre et types de pièces très limités et non remplaçables pour le Robot mBot V1.1
- Non évolutif

Notre second concurrent est la marque LEGO. Elle propose des Kits de programmation de robots qui sont plus accessibles aux enfants.

Voici deux de leurs kits :

LEGO (Mindstorm NXT 2.0)

LEGO (Mindstorm EV3)

Avantages

- Logiciel de programmation simplifié
- Facile à assembler

Inconvénients

- Capteurs et moteurs propriétaires cher à l'achat
- Puissance des actionneurs limitée
- Possibilité limitée du nombre de capteurs (4) et d'actionneurs (4) sur un même robot à cause du nombre de ports disponibles.

Nous allons alors, pour notre Kit, nous inspirer des avantages de ces concurrents et essayer de retirer les inconvénients. Nous avons ensuite élaboré un scénario d'usage de notre Kit pour enfin passer à l'élaboration du cahier des charges.

1.4 Scénario d'usage

Nous avons adapté le scénario d'usage du Wiki pour qu'il soit en adéquation avec les résultats finaux de notre projet.

Hervé est en Peip à Lille et il s'est découvert une nouvelle passion pour la Robotique. Il a vu plusieurs reportages et vidéos sur le sujet. Il veut travailler plus tard dans ce domaine. Il décide alors qu'il ira en IMA . En attendant, il décide d'apprendre toutes les facettes de cette discipline. Mais il ne sait pas par quoi commencer. Il veut apprendre étape par étape et se renseigne alors sur des moyens d'apprendre la robotique facilement. Sur internet il découvre l'existence de Kit de robotique programmable qui constitue une excellente base de Robotique et qui sont accessibles aux débutants. Mais ils sont assez chers et il n'a pas les moyens (il est étudiant).

Il découvre alors l'existence du Club "Robotech". Quelle chance ! Les membres du club l'informent de l'existence d'un tout nouveau Kit crée par des étudiants pour leur projet de 4ème année.

Hervé se renseigne alors grâce à leur Wiki : ce kit est Open Source et il pourra grâce à l'imprimante 3D et la découpeuse Laser du Fab fabriquer les pièces de son Kit. Sur le Wiki, les élèves ont expliqué toutes les étapes de réalisation du Projet. Hervé pourra alors comprendre en détail la conception du Kit. Cela lui permettra également de faire des économies d'argent. Pour ce qui est des capteurs et de la carte, les membres de Robotech (qui sont curieusement principalement des IMA) lui fournissent de l'aide et il pourra tout réaliser à Polytech.

Il sait que réaliser toutes les parties de son Kit (cartes électroniques, pièces ..) sera assez long mais il pourra avancer à son rythme et réaliser son Robot étape par étape. De plus, il gagnera un temps précieux en programmation car les élèves ont tout prévu : programmation par blocs, librairies disponibles pour faciliter la programmation sur l'IDE Arduino, et application Bluetooth.

Quelques semaines plus tard, Hervé réalise son premier robot qui est commandable en Bluetooth et possède toute une panoplie de capteurs lui proposant un champ d'application très varié. Il est également très content de la batterie en Lithium du Kit qui assure une bonne autonomie à son robot et qui se recharge très facilement.

Malheureusement il a fait tomber son robot (il est maladroit) et a cassé des pièces en plexiglas. Pas de soucis ! Grâce aux fichiers disponibles sur le Wiki, il peut en fabriquer un nouveau facilement.

Au fur et à mesure, Hervé utilise de nouveaux capteurs et fabrique de nouvelles pièces pour améliorer son robot. Grâce à ce Kit il a pu apprendre à programmer son propre robot et comprendre comment sont fabriquées les différentes parties et leur fonctionnement.

Un an plus tard, le voilà nommé Secrétaire et Responsable de la communication de Robotech. La date de la compétition phare du club, la Coupe de France de Robotique, approche. Surchargés de travail entre les cours de réseau et d'électronique, les membres du club n'ont pas eu le temps de fabriquer le robot secondaire et il ne leur reste qu'un mois. Hervé décide alors, sur un éclair de génie, d'utiliser de nouveau le Kit : celui-ci est robuste et permettra de confectionner n'importe quel type de robot. L'avantage est qu'il est facilement programmable via l'IDE Arduino pour les jeunes IMA3, et on pourra y brancher énormément de capteurs sans un nombre incalculable de fils qui se prendront dans les roues !

1.5 Cahier des charges

- Créer une carte programmable optimisée permettant le branchement facile des capteurs et actionneurs sur un ATMega2560
- Créer et/ou modifier des capteurs pour le robot pour être adapté à l'I2C
- Concevoir des librairies pour faciliter la programmation sur l'IDE Arduino
- Proposer des modèles et une programmation simplifiée par blocs
- Concevoir les pièces 3D du robot grâce aux imprimantes 3D

- Concevoir les pièces 2D du robot dans un matériel résistant, peu cher et facile à usiner : plexiglass
- Concevoir une application Bluetooth pour contrôler le Robot
- Faciliter la construction (code couleur, câbles simples à brancher)

1.6 Liste des tâches à effectuer

Tâche 1 :**Création de la carte principale** Après en avoir parlé avec nos enseignants, nous utiliserons comme base de notre carte programmable, un Arduino Mega.

Tâche 2 :**Création carte moteurs et capteurs** Pour pouvoir contrôler le robot, on peut utiliser différents capteurs et des moteurs.

Tâche 3 :**Création des pièces** Créer les pièces du Kit pour pouvoir créer son robot selon son besoin.

Tâche 4 :**Programmation** Nous allons créer des librairies pour faciliter la programmation des différents modèles. La programmation se fera avec l'IDE Arduino (langage C) ou avec de la programmation par blocs. Nous voulons aussi réaliser une application Bluetooth pour commander le Robot via SmartPhone.

2. Réalisation du Projet

Après avoir réalisé la liste des tâches à effectuer, nous avons réalisé un calendrier prévisionnel afin d'organiser notre projet dès le début. La gestion de projet est très importante, nous avons également dressé un tableau d'avancement des objectifs. Notre Wiki permet de voir l'avancement du projet semaine après semaine.

2.1 Choix technique : matériel et logiciel

Pour les pièces du Kit nous allons utiliser du plexiglas de 3cm et 6cm d'épaisseur pour avoir des pièces solides. Nous utiliserons aussi l'imprimante 3D pour certaines pièces. Nous avons besoin également de vis M4 de 12mm et 16mm et d'écrous. Les vis M4 aurons l'avantage d'être plus facile à manipuler pour les plus jeunes et de rendre la structure plus solide que des M3.

Pour la partie Programmation, nous utilisons l'IDE Arduino pour programmer le Kit. C'est la manière la plus répandue pour coder sur Arduino. Nous allons également créer des librairies qui seront disponibles dans un GitHub. La programmation pourra se faire en bloc grâce à Blockly Arduino, un outil de programmation en blocs sur navigateur plus agréable à utiliser qu'Ardublock. Enfin la création de l'application Bluetooth sur Android destinée à contrôler le robot sera faite sur l'application Appinventor pour un gain de temps**.**

Pour la partie Électronique, nous avons besoin de beaucoup de matériel pour que le Kit soit complet. Nous choisissons 2 moteurs encodeurs avec assez de couple pour tracter notre robot. L'avantage d'un encodeur et de pouvoir réguler le moteur à courant continu en position. Ils sont également moins cher et moins encombrant qu'un moteur pas à pas. Nous pourrons donc créer des fonctions d'odométrie comme "avancer de 10 centimètres" par exemple.

Nous avons également besoin de divers capteurs. Nous choisirons deux sortes de capteurs de distance (Sharp pour les distances proches, Ultrasons pour les obstacles éloignés), des capteurs de fin de course pour le bras que nous construirons pour commander la pince et pour finir des capteurs de lignes.

Pour la connectique nous choisissons les câbles RJ25 qui ont l'avantage d'être facile à connecter pour simplifier le câblage. Nous avons également confectionné nous même les fils avec les connecteurs JST. Pour cela nous nous sommes minis de fil, d'une pince à sertir et les connecteurs souhaités.

Concernant la batterie, nous nous sommes orientés vers des batteries Lithium 18650 3.7V. Ces batteries ont une excellente autonomie pour notre application, et une fois 2 batteries placées en série, nous obtenons une tension de sortie de 7,4V et un courant maximum de 5A. Au vu de la consommation de nos moteurs, cette solution est idéale et prend peu de place sur le robot comparé à 6 piles AA 1.5V.

Après cela chaque carte électronique nécessite de nombreux composants spécifiques à sa conception. La liste détaillée est résumée dans le Wiki.

2.2 Partie Électronique

Pour cette partie nous avons décidé d'utiliser le logiciel de programmation **Eagle** qui, même si il nous ne l'avons jamais utilisé contrairement à Fritzing, est plus performant pour notre application et est utilisé dans les entreprises de nos jours. Nous créons en premier le schematic des cartes, puis nous routons la carte en respectant les règles établies :

- éviter les angles droits
- avoir un plan de masse bien réparti
- faire les pistes les plus courtes possibles
- bien espacer les pistes
- pour les signaux à fréquence élevée, les angles droits sont à proscrire
- isoler la partie régissant l'horloge des microcontrôleurs (Quartz) des autres pistes pour éviter les perturbations.
- prendre garde à la taille des pistes, plus le courant qui doit passer dedans est élevé, plus les pistes doivent être larges
- rajouter des trous pour pouvoir fixer la carte

Tous les Schematics et les Board des cartes électroniques sont disponibles sur le Wiki.

2.2.1 La carte principale

Nous voulons créer la carte principale à base d'Arduino Mega. Au départ, nous voulions nous inspirer des cartes Arduino créés dans les BE PEIP.

Néanmoins comme nous voulons comme base un Arduino Mega, nous nous sommes inspirés de schémas sur internet: Schematic Arduino Mega (open source). Le choix d'un ATMEGA2560 plutôt qu'un 328p s'est fait sur le nombre d'entrées et sorties, et d'interruptions que nous voulons utiliser pour notre kit. En effet un Atmega328p dispose d'un nombre d'I/O réduit.

La carte principale comprendra plusieurs éléments:

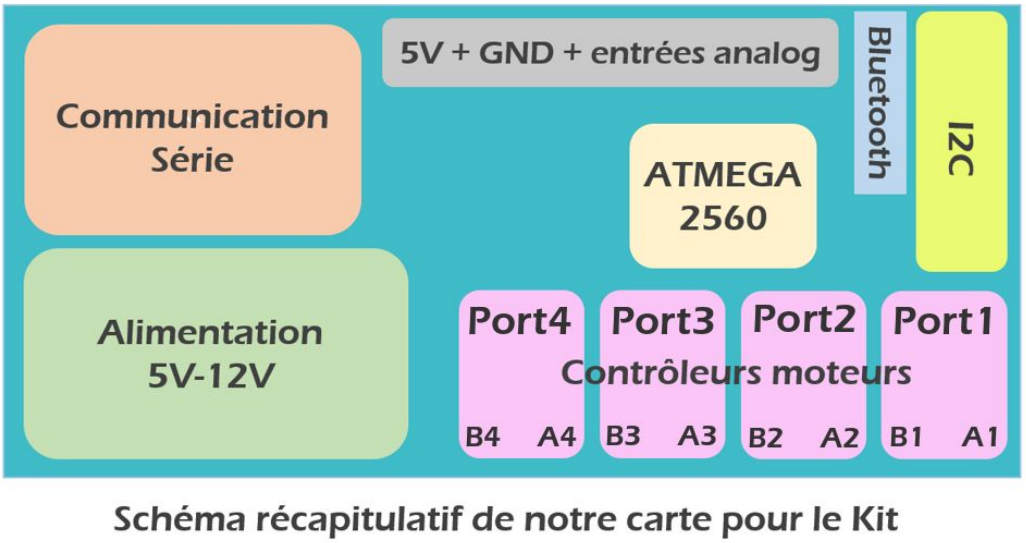


Figure 1 : schéma récapitulatif des différentes parties de la carte Principale

Cette carte sera la plus difficile à réaliser. Il faut donc être méthodique et définir les éléments étapes par étapes. Il faut étudier les différents entrées et sorties de la carte qui seront nécessaires.

Pour avoir l'empreinte de certains composants, il nous faut rechercher et/ou créer des librairies pour les composants à utiliser. Cette étape a été bien plus fastidieuse et longue que prévu.

Nous pouvons récupérer certaines empreintes sur les sites des fournisseurs de composants électroniques (Mouser, RS, etc...). Pour les autres nous nous basons sur les schémas et les cotations fournies par leur datasheet.

Le routage ne fut pas évident : en effet, il y a un grand nombre de pistes et une contrainte de l'emplacement des contrôleurs moteurs. Il a fallu plusieurs fois recommencer certaines parties à cause de piste que nous n'arrivions pas à placer. C'est un travail de patience où il faut anticiper les tracés que nous effectuons.

Nous choisissons de laisser quelques pins à disposition sur la carte pour rendre le Kit évolutif. Il est donc possible à l'avenir d'ajouter des éléments plus évolués.

Nous expliciterons plus en détails les problèmes rencontrés pour cette carte dans la partie "Présentation des résultats et remarques".

Communication série de la carte:

Pour la partie communication série nous n'avons aucune amélioration par rapport à un arduino Mega classique, nous reprenons donc cette partie pour notre carte principale en prenant tout de même soin de bien comprendre son fonctionnement.

Alimentation de la carte

Nous souhaitons pouvoir alimenter la carte entre 6V et 12V afin d'alimenter directement les moteurs, sans sources extérieur.

Les moteurs ayant une consommation sous 6 Vcc à vide de 0,1 A et bloqué de 2,7 A nous nous orientons sur une solution permettant de transmettre assez de courant.

Cette alimentation doit pouvoir être transformer de manière dynamiquement en 5V pour l'alimentation des contrôleurs et des capteurs. En effet la tension des batteries lithium que nous utiliserons a tendance à varier avec son état de charge. Le composant doit donc être capable de transformer n'importe quelle tension en 5V constant.

La partie 5V doit débiter assez de courant pour pouvoir alimenter tous les capteurs que nous brancherons dessus ainsi que potentiellement des servomoteurs si l'utilisateur final souhaite en rajouter.

La solution retenue est la suivante: Pour la partie alimentation des moteurs, celle ci sera reliée directement à l'alimentation générale. Nous n'oublions pas d'inclure un interrupteur permettant de couper facilement l'alimentation de la carte via les batteries lithium. Le port d'entrée se fera par un port Jack 5.5mm pour faciliter le branchement.

Pour la partie convertisseur vers 5V nous avons choisi le composant : TPS54331

Ce dernier est un Step-Down DC/DC permettant sur une alimentation de 3.5V à 12V de ressortir une tension de 5V sous une limite de 3A suffisante pour notre application. En parcourant la data sheet du constructeur, il est alors facile de reconstituer le circuit grâce aux formules et exemples fournis. Nous y ajouterons un fusible réarmable adapté et une diode pour protéger le circuit. Un transistor est présent pour pouvoir basculer en priorité sur l'alimentation sur batterie quand le port USB et la batterie sont tous deux branchés

VIN (V)	VOUT (V)	f_{SW} (kHz)	L_o (µH)	C_o	R_{O1} (kΩ)	R_{O2} (kΩ)	C_2 (pF)	C_1 (pF)	R_3 (kΩ)
12	5	570	6.8	Ceramic 33 µF, ×2	10	1.91	39	4700	49.9
12	3.3	570	6.8	Ceramic 47µF, ×2	10	3.24	47	1000	29.4
12	1.8	570	4.7	Ceramic 100 µF	10	8.06	68	5600	29.4
12	0.9	570	3.3	Ceramic 100 µF, ×2	10	80.6	56	5600	29.4
12	5	570	6.8	Aluminum 330 µF, 160 mΩ	10	1.91	68	120	29.4
12	3.3	570	6.8	Aluminum 470 µF, 160 mΩ	10	3.24	82	220	10
12	1.8	570	4.7	SP 100 µF, 15 mΩ	10	8.06	68	5600	29.4
12	0.9	570	3.3	SP 330 µF, 12 mΩ	10	80.6	100	1200	49.9

Figure 2 : Datasheet du composant

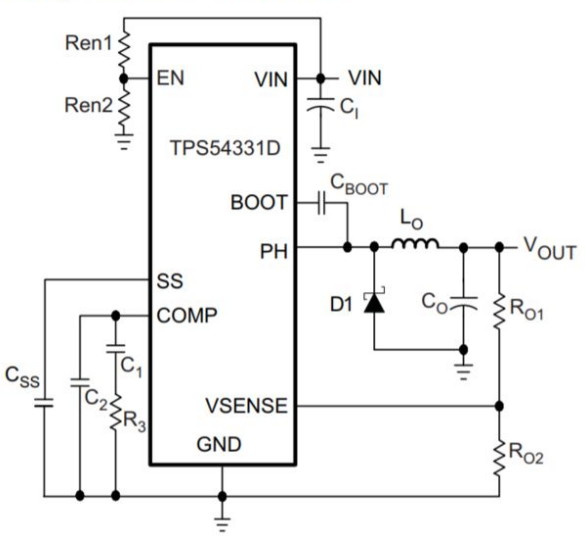


Figure 3 : Schéma simplifié de la datasheet

(voir Annexe 1 : Carte Principale)

2.2.2 Les cartes capteurs

Les cartes capteurs seront connectées entre elles grâce au protocole I2C et à des câbles RJ25 pour simplifier le branchement.
Ce protocole est très utilisé en Robotique. Il permettra de simplifier le montage du Robot en limitant le nombre de fils à connecter.

L'I2C est un bus permettant d'échanger des données entre plusieurs systèmes numériques. Généralement, ce type de bus sert à connecter un micro-contrôleur (notre carte principale) à plusieurs périphériques comme nos capteurs ou des actionneurs.
Ce bus se compose de 2 fils :

- **SDA** : qui sert à transmettre les données, c'est par cette ligne que transitent les informations échangée entre les deux systèmes.
- **SCL** : qui sert à transmettre une horloge permettant la synchronisation entre les 2 systèmes échangeant des données.

Nous avons commandé plusieurs capteurs :
- Gyroscope (capteur de position angulaire)
- Suiveur de ligne
- Fin de course
- Capteur de distance Ultrasons
- Capteur de distance par infrarouge Sharp

Chaque carte réalisé sous Eagle comprendra:

- une entrée et une sortie: la connexion entre les cartes se fera par des câbles RJ25 afin de disposer d'un système de branchement facile.
- l'ATtiny85 (pour les capteurs tout ou rien)ou ADS (pour les capteurs analogiques)

- un ensemble de 6 pins afin de pouvoir la connecter à la carte principale et pouvoir programmer l'Attiny en ICSP
- Des pins JST pour pouvoir connecter facilement un capteur
- Des capacités de découplage
- Des résistances Pull-Up de 10k sur SCL et SDA

Nous avons choisi d'utiliser des ATtiny85 pour la partie I2C. L'ATtiny85 est un microcontrôleur d'Atmel (même famille que l'ATmega des Arduino). Ce composant est plutôt facile à inclure dans un projet, il est compacte et peu cher.

Les ATTiny ne disposent pas à proprement parler d'une stack I2C mais intègrent un USI (Universal Serial Interface) qui peut se programmer pour communiquer en I2C.

L'avantage d'utiliser un microcontrôleur plutôt qu'un composant dédié purement à l'I2C avec des entrées digitales, est de pouvoir utiliser des capteurs avec un fonctionnement un peu plus complexe qu'un fin de course. Par exemple, pour un capteur Ultrasons nous avons besoins d'émettre une onde sonore, de mesurer le temps que met l'onde pour atteindre un objet puis de rebondir sur lui et de revenir. Avec un ATtiny85 nous pouvons alors inclure les étapes de calcul dans le contrôleur et directement communiquer la distance mesurée en I2C.

Nous utiliserons l'ATtiny85 en CMS par soucis d'encombrement (pour prendre moins de place).

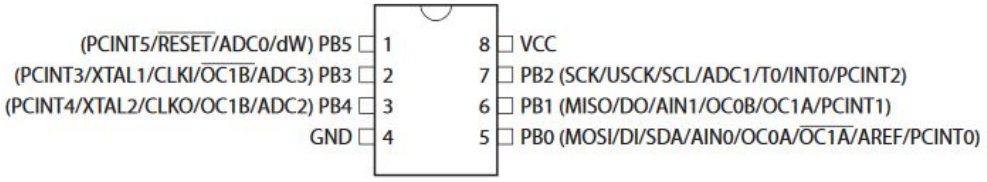


Figure 4 : ATtiny85

Nous pensions faire une carte différente pour chaque capteur contrôlé par ATtiny85 car nous pensions les faire imprimer à Polytech. Il s'est avéré que la réalisation de ces cartes à Polytech était impossible en raison des prises RJ25 et des vias. Nous avons donc finalement fait une unique carte ATtiny85 commune pour tous les capteurs. Nous les avons produites en plusieurs fois et il suffit de souder les pins nécessaires au capteur voulu.

Le capteur SHARP, quant à lui nécessite, un ADC (analogic to digital). Nous n'utilisons donc pas un ATtiny85 mais un ADS1015 (CMS) qui contient une interface I2C. Les capteurs Sharp auront une carte adaptée.

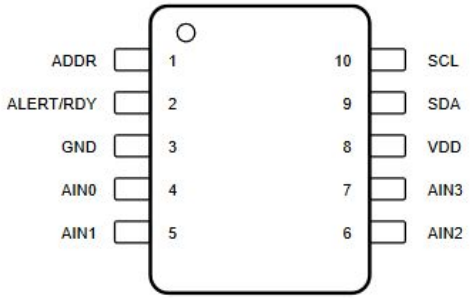


Figure 5 : ADS1015

L'ADS1015IDGST est un convertisseur de précision Analogique-Numérique (ADC) avec une résolution de 12 bits. Il a une faible consommation en courant. Il comporte une référence et un oscillateur intégrés. Les données sont transférées via une interface série compatible I2C, quatre adresses esclaves I2C peuvent être sélectionnées grâce au pin ADDR. Il fonctionne avec une alimentation unique allant de 2,0 à 5,5V. Voici les différentes adresses que peut prendre l'ADS1015 en reliant le pin ADDR à un autre:

- ADDR to Gnd: address = 0b01001000 = 0x48 = 72
- ADDR to VDD: address = 0b01001001 = 0x49 = 73
- ADDR to SDA: address = 0b01001010 = 0x4A = 74
- ADDR to SCL: address = 0b01001011 = 0x4B = 75

Finalement, le gyroscope communique avec un microcontrôleur via un port I2C. Nous avons donc décidé de ne pas faire de carte pour ce capteur car il n'en a pas besoin.

Nous avons donc 2 types de cartes : ATtiny85 et ADS1015. Nous avons partagé les fichiers Schematics et Board sur le Wiki (voir Annexe 2 : Cartes Capteurs)

2.2.3 Les cartes contrôleurs moteur

La partie la plus importante à prévoir est l'emplacement des pins des contrôleurs moteurs, il a donc fallu effectuer des recherches pour commencer le schematic. Nous listons alors les différents pins dont nous aurons besoin pour notre contrôleur moteur afin de les disposer sur notre carte principale. Nous avons alors besoin pour les contrôleurs de :

- 8 entrées:
  - ➢ 2 pins PWM: réguler la vitesse des 2 moteurs
  - ➢ 1 pin d'interruption: pour l'encodeur du moteur
  - ➢ 4 sorties digitales: pour contrôler le sens de rotation de chacun des 2 moteurs
  - ➢ 1 pin en plus pour brancher un contrôleur de moteur pas à pas dans le futur (évolutivité du kit)
- 8 sorties:
  - ➢ +5V : alimentation de l'encodeur
  - ➢ +Vcc: pour l'alimentation des moteurs
  - ➢ 2 GND
  - ➢ les 4 sorties des moteurs

Concernant la réalisation de la carte, la principale contrainte était la dimension du contrôleur moteur. Nous voulions qu'il soit assez compacte pour ne pas encombrer la carte principale et que l'emplacement des pins soit compatible avec d'autres contrôleurs du commerce afin de rester dans l'esprit d'évolutivité du kit.

Niveau composant électronique nous utilisons un contrôleur (de type pont en H) de type TB6612 auquel nous rajoutons des capacités de découplages et un transistor de protection contre les retours de puissance. Ce contrôleur peut délivrer jusqu'à 1,2A en continu et 3,2A en impulsionnel pour une alimentation maximale de 15V. Il est donc parfaitement adapté aux moteurs que nous avons sélectionnés. L'avantage est également d'avoir un composant CMS compacte pour le problème de place..

Le contrôleur peut piloter 2 moteurs à la fois, soit 2 moteurs DC: l'un sur le PortA et l'autre sur le PortB. Soit 1 moteur encodeur qui se branche sur le PortB, le PortA est donc disponible pour un moteur DC. (voir Annexe 4 : Cartes Contrôleurs moteur)

2.3 Partie Programmation

Nous avons voulu simplifier la programmation du Robot au maximum. En effet, nous voullons que notre Kit soit accessible aux novices en programmation.

2.3.1 ATtiny85

Pour faire fonctionner la carte des capteurs utilisant l'ATtiny85 nous devons créer le programme de communication I2C que nous lui téléversons. L'objectif est de programmer l'ATtiny85 en "Escale" I2C. L'arduino en mode "Maître" envoie un octet à l'adresse de l'esclave pour lui indiquer de quel capteur il souhaite la mesure puis il lui envoie une requête pour recevoir la donnée.

Comme dit précédemment, l'ATtiny85 ne possède pas de réelle interface I2C, heureusement plusieurs librairies Open Source existent sur Internet pour faciliter cette démarche. Mais beaucoup ne sont plus à jour où fonctionnent avec un retard important. Après plusieurs tests nous avons trouvé cette [librairie] qui a l'avantage de pouvoir fonctionner par interruption. Il y donc une meilleure gestion du bus I2C et une réduction de la consommation de courant.

Une fois ce programme de communication établi nous intégrons les différentes routines de mesure sur les capteurs. Nous optimisons les données en transmettant la valeur du capteur demandé sous la forme d'un seul octet pour éviter les erreurs de transmission. Nous réservons la valeur 0xFF, qui servira de message d'erreur, indiquant la présence d'un problème sur le bus.

Il faut savoir que l'ATtiny85 n'est pas directement programmable via un port USB. Il a fallu donc le programmer grâce à un Arduino configuré en ISP.

2.3.2 Les librairies

Nous avons voulu créer des librairies pour les composants à utiliser. Elle est codée en C++ et est constituée d'un ou plusieurs fichiers .h où seront déclarés les classes ainsi que le fichier .cpp où seront explicitées les fonctions des classes.

De nombreux exemples doivent également être inclus pour guider les utilisateurs dans le maniement de la librairie.

Nous avons fait le choix de diviser notre librairie en plusieurs fichiers .ccp et .h :

- **Driver_moteur** : cette partie déclare la classe représentant les moteurs encodeurs. Ses attributs sont entre autre les pins sur lesquels il est connecté, les paramètres mécaniques du moteur et les différentes valeurs déduites des encodeurs: vitesse réelle/voulue, nombre de pulsations, etc…

Parmi les fonctions, on y trouve trois modes de mouvements différents:

  - ➢ Contrôle du moteur par PWM
  - ➢ Contrôle du la vitesse en rpm du moteur. Ici on implémente un régulateur fonctionnant grâce aux valeurs reçues de l'encodeur. Un simple régulateur P suffira, nous n'avons pas besoin d'être particulièrement précis sur la vitesse du moteur.
  - ➢ Contrôle du moteur en position. On y trouve là l'intérêt principal de l'encodeur sur un moteur à courant continu. On peut ainsi contrôler l'angle de l'arbre moteur (et des roue) avec un moteur bien plus léger et moins volumineux qu'un moteur pas à pas. Ici on implémente non pas un simple régulateur P mais un régulateur PD qui permettra d'atteindre un angle donnée de la manière la plus optimisée possible.

Nous avons dû effectuer de nombreux tests pour trouver les paramètres des régulateurs.

- **MotorDC**: cette partie est dédiée au contrôle des moteurs à courant continu simples grâce aux contrôleurs moteurs. On y déclare la classe MotorDC mais aussi deux classes héritières pour la pince et le bras. Pour ces dernières on a alors des fonctions plus parlantes comme "open()" et "close()" ou encore "up()" et "down()". Elles ont comme attributs les pins sur lesquels le moteur est branché.
- **Sensors_kit**: On définit ici les classes pour chaque type de capteur et les fonctions permettant de recueillir leurs valeurs.
- **MyRobot**: la classe présente dans cette librairie représente le robot où on lui attribuera un moteur gauche, un moteur droite, des capteurs, etc... On y renseignera également la largeur entre les roues du robot ainsi que leur diamètre. Ces valeurs permettent alors de créer des fonctions d'odométrie comme "avancer de 10cm" ou encore "tourner à droite de 90 degrés". On y trouvera également une fonctionnalité pour être contrôlé en Bluetooth.

A chaque classe ainsi créée nous avons écrit un programme d'exemple pour montrer l'utilisation de chaque fonction.

Le développement de la librairie a pris énormément de temps, le plus long a été la partie sur l'encodeur qui possède un fonctionnement bien particulier. Il a fallu tester en permanence les programmes et les corriger en conséquence.

Il ne faudra également pas oublier d'installer la librairie correctement comme décrit sur le GitHub. **Lien du repository GitHub de la librairie**.

Au final, la création de la librairie Arduino a été bien plus fastidieuse et longue que prévu. Elle permettra de programmer plus facilement notre Robot.
(voir Annexe 5 : Programmation)

Nous avons effectué des tests sur la carte ATtiny85. L'objectif de test est de récupérer la valeur d'un bouton connecté sur l'Attiny et d'envoyer sa valeur sur requête de l'arduino en I2C. L'arduino est connecté en tant que Master et l'Attiny en tant que Slave. Nous avons trouvé la librairie suivante[librairie ATtiny] qui a l'avantage de pouvoir fonctionner par interruption pour pouvoir faciliter le test.

Dans un premier temps, l'ATtiny85 n'est pas directement programmable via un port USB. Nous réalisons donc, en s'inspirant de ce tutoriel **[Tutoriel]**, un shield pour Arduino UNO permettant facilement de programmer l'ATtiny traversant avec des Leds permettant de visualiser lorsque le téléversement du programme est en cours où qu'une erreur est survenue. Coté arduino, il suffira de lui téléverser l'exemple "ArduinoISP". Puis pour téléverser le programme il suffit alors de connecter l'ATtiny sur le shield, et de sélectionner le profil ATtiny85 et le profil de téléversement "Arduino as ISP" sur l'IDE.

Tout est maintenant réuni pour effectuer les tests. Nous avons expliqué les tests en détails sur le Wiki. Tous les tests sont concluants, nous obtenons le résultat souhaité.

2.3.3 La programmation par blocs (Blockly Arduino)

Afin de faciliter la programmation du Kit, nous avons créé des blocs Blockly Arduino.
Nous avons finalement préféré utiliser Blockly Arduino plutôt que ArduBlock.
Blockly Arduino est une application web qui ne nécessite aucune installation (pas besoin de droits administrateur non plus) et qui fonctionne sur toutes les plateformes (PC, Mac, Androïd, …).

Un autre avantage de Blockly Arduino est qu'il est plus récent. En effet, Ardublock n'est plus vraiment maintenu et activement développé.

L'utilisation de Blockly Arduino permettra à l'utilisateur du Kit de travailler sur ce qui se fait de mieux niveau programmation en blocs. Il a la particularité d'avoir une interface de travail plus accueillante et surtout d'être ouvert à l'ajout de nouvelles librairies de blocs.

(voir Annexe 5 : Programmation)

2.3.4 L'application Bluetooth

La création d'une application Bluetooth permet de contrôler facilement le robot en mode manuel. N'ayant pas de connaissances préalables sur le développement d'applications mobiles et surtout de communication en Bluetooth, Appinventor est un outil parfait pour les débutants et permet d'économiser du temps. De plus, il propose des outils simplifiés pour la communication en Bluetooth.

Après s'être familiarisé avec le logiciel, nous avons programmé une application de base permettant de transmettre un octet à un module Bluetooth HC-05. Il faut maintenant choisir les fonctionnalités de l'application vis à vis du robot. Nous avons décidé d'implémenter les fonctionnalités de base du robot, à savoir avancer, reculer, tourner à gauche, tourner à droite, monter le bras, descendre le bras, ouvrir la pince et fermer la pince. Un "slider" permet de régler la vitesse du robot.

Pour l'envoi des données nous nous sommes accordé sur l'envoi des données sous 1 seul octet. Ce format a pour avantage de limiter les erreurs causées par l'envoi de plusieurs octets et améliorer le temps de réponse du robot. Nous disposons de 9 boutons, nous avons donc besoin de 4 bits pour coder le bouton appuyé. Ils se trouveront sur les bits de poids faible. Il reste donc les 4 bits de poids fort pour encoder la vitesse soit 15 valeurs possibles. Il faut donc limiter la plage de vitesse autorisées. Nous avons choisi une sélection de la vitesse par pas de 10% (min=10%, max=100%) donc 9 valeurs différentes.
Par exemple: 0xA2 = 0b1010 0010 demande au robot d'avancer à la vitesse 100%
Apk de l'application mobile Androïd: **Fichier:ApkRobotKit.zip**

L'application est téléchargeable sur Smartphone via un QR Code (voir Annexe 5 : Programmation)

2.4 Partie Modélisation et Mécanique : Les pièces du Kit

Pour construire un Robot avec le Kit, il est indispensable d'avoir un grand nombre de pièces. Nous avons décidé de créer plusieurs types de pièces afin de permettre à l'utilisateur de construire son propre Robot. En effet, l'objectif de tout Kit de Robot est de permettre la construction de n'importe quel type de Robot.

Toutes les pièces proposées sont assemblables avec les vis M4 de 12mm/16mm et les écrous M4 du Kit. Voici comment nous nous sommes organisés:

2.4.1 Les pièces 2D

Le Fabricarium possède une découpeuse laser. Elle permet de découper beaucoup de types de matériaux. L'avantage d'utiliser la découpeuse laser est le gain de temps. son utilisation est plus rapide que d'imprimer avec l'imprimante 3D. Afin d'avoir des pièces solides nous avons décidé d'utiliser du plexiglas. Nous utilisons 2 types d'épaisseurs différents : 3mm et 6mm.

Avec le plexiglas de 6mm, nous avons créé des pièces plus ou moins longues possédant des trous adaptés aux vis. Elles servent actuellement de châssis pour le Robot. Nous avons également imprimé des pièces d'angles et un support moteur DC avec le plexiglas de 3mm.

2.4.2 Les pièces 3D

Grâce à l'imprimante 3D, nous avons pu modéliser de nombreuses pièces essentielles à notre Kit. Nous les avons modélisé sur Fusion360, Onshape ou encore Tinkercad.

- Pince et son bras

Nous avons modélisé la pièce dans laquelle on viendra glisser le moteur du bras, ainsi que les engrenages qui entraînent son mouvement et donc faire monter ou descendre ce dernier. La pince également modélisée en 3D possède plusieurs pièces qui viennent s'assembler pour permettre le mouvement de fermeture et d'ouverture. Elle est couplée à un moteur à courant continu dont la particularité est de posséder un axe fileté de type M4. La rotation du moteur entraîne alors le mouvement d'un écrou que nous avons glissé dans une pièce lors de son impression pour qu'elle soit piégée dedans. Nous parvenons ainsi à motoriser le processus de fermeture et d'ouverture de la pince.

- Maillon Chenille et roues correspondantes

Cette chaîne sera composée de différents maillons que nous pourrons assembler pour modifier la longueur de celle-ci. Pour l'assemblage, il suffira juste de glisser un morceau de filament de 1.75mm d'imprimante 3D dans les maillons pour les assembler entre eux.

Pour l'adhérence, il faut rajouter une bande de colle chaude au pistolet à colle.

Nous modélisons également les roues spécifiques qui pourront entraîner la rotation de la chenille. Deux sortes de roues sont nécessaires. La première possède un méplat pour se placer sur le moteur. L'autre sera la roue non motorisé que l'on viendra maintenir avec une vis sans trop la serrer pour ne pas contraindre sa rotation.

- Roues sans chenilles (moteur et libre)

Pour l'adhérence, il faut rajouter soit une bande de colle chaude au pistolet à colle, soit des élastiques.

- Support de capteur et moteur

Nous avons créé des boitiers pour les capteurs afin de les protéger des chocs et de les placer facilement sur le robot. Pour le moteur il y a une pièce pour fixer les moteurs encodeurs ainsi qu'une seconde pièce pour maintenir le moteur droit puisqu'il avait tendance à pencher sous le poids du robot.

- Support de carte électronique

Nous créons un support adapté aux trous créés sur la carte électronique. Nous pouvons ainsi plus facilement la fixer sur notre robot.

- Support de batteries

Il permet d'accueillir les batteries 18650 du kit.

- Pièces de coin et d'angle

Elles permettent par exemple de réaliser le châssis du robot. Les angles et coin facilitent le montage du Kit

- Boîte à vis (couvercle + fond) + Clé anglaise d'aide au montage

Petit bonus pour le rangement des vis :)

Pour présenter toutes ces pièces, nous avons créer un catalogue (voir Annexe 6 : Catalogue des pièces du Kit)

2.4.3 Le robot

Nous avons choisi de réaliser un Robot chenille + pince comme modèle de base, c'est un modèle complet et qui sera le plus utile pour la Coupe de France. En effet, les règles de cette Coupe changent chaque année mais l'objectif est souvent de saisir puis de déplacer des objets, d'où l'utilité de la pince et du bras. Les chenilles permettent de rendre le robot tout terrain et facilement contrôlable. Mais d'autres pièces comme des roues simples sont également à disposition pour créer d'autres formes de robots selon l'imagination de l'utilisateur. Les pièces ont été imprimées dans un nombre limité d'exemplaire mais tous les plans sont disponibles si jamais il vient à en manquer.

(voir Annexe 7 : Evolution du Robot)

3. Présentation des résultats et remarques

3.1 Problèmes rencontrés

Carte Principale

La carte principale nous a pris beaucoup de temps à cause de plusieurs problèmes rencontrés.

➢ Certains composants n'étaient plus en stock entre le début du routage et le moment d'envoyer la commande. Il a donc fallu en choisir d'autres et redessiner en urgence par nous même leur empreinte pour la carte, notamment pour les interrupteurs donc l'empreinte n'est pas standardisée. Pour cela nous avons dû créer notre propre librairie contenant les composants dessinés en se basant sur les Datasheets constructeur.

➢ Lors de la soudure des composants nous nous sommes rendus compte que la librairie du contrôleur de puissance récupérée sur le site du fournisseur était erronée. Des pins étaient inversés. Pour palier à ce problème nous avons fait imprimer à Polytech la partie servant à transformer la tension batterie en 5V. Cette partie viendra se souder sur les pins de la carte par dessus l'ancienne partie. Malheureusement, la partie contrôleur de puissance ne fonctionne toujours pas et nous avons manqué de temps pour nous y intéresser plus longuement.

➢ Lors des premiers tests de la carte principale, nous avons réussis à installer le bootloader sur l'Atmega8u2. Cependant l'ATMega2560 ne répondait pas.

Après de longues heures de test nous avons découvert que le Quartz de l'ATMega2560 n'oscillait pas. Nous l'avons donc changé avec un nouveau. Mais la carte ne fonctionnait toujours pas et avec nos encadrants nous ne comprenions pas le problème. Mr Redon a proposé alors 2 solutions:

- soit le contrôleur n'a pas résisté à la soudure au four
- soit certaines soudures sur l'ATMega ne seraient pas assez prononcées.

Après avoir renforcé les soudures le contrôleur répondait enfin. Nous avons pu téléverser le bootloader de la carte et téléverser un premier programme Blink sur la Led13 pour vérifier son bon fonctionnement.

➢ Une erreur s'était glissée dans la carte et des pistes n'ont pas été reliées. Nous avons donc soudé un strap pour remédier au problème. Cependant malgré le strap nous ne pouvions toujours pas téléverser de programme via la liaison série entre le 8u2 et le 2560. Nous avons essayé de dessouder le réseau de résistance de cette liaison série qui semblait défectueux et d'y souder à la place de nouvelles résistance disposées sur la tranche pour entrer dans l'empreinte du réseau. Mais le problème n'était toujours pas résolu, de plus l'une des led témoins de transfert à été endommagé lors du dessoudage du réseau avec le pistolet à air chaud qui était indispensable pour retirer ce composant. Nous avons vérifié une fois de plus au multimètre les soudures, tout semblait pourtant normal.

➢ Malheureusement par manque de temps nous ne pouvions plus nous permettre d'essayer de réparer les parties non fonctionnelles de la carte. Nous avons préféré nous concentrer sur les parties qui fonctionnaient et utiliser la carte du mieux que nous le pouvions.

La partie convertisseur de puissance ne fonctionnant pas, nous avons dû alors utiliser 2 alimentations en attendant. Pour contourner la liaison série, nous téléversons les programmes en ICSP. La batterie Lithium servira à alimenter les moteurs alors que l'alimentation du 5V se fera via un câble USB sur Ordinateur ou via une batterie portable externe (celles qui permettent de recharger nos téléphones portable).

Nous avons positionné alors la carte principale sur le robot et avons testé les moteurs directement dessus. Le contrôle s'est correctement déroulé et la librairie Arduino fonctionne bien. Nous avons enfin réussi à piloter notre robot en Bluetooth avec notre propre carte électronique et non pas avec un Arduino du commerce.

Autres

➢ Nous avions décidé de contrôler nos deux capteurs Sharp sur la même carte. Nous avons finalement préféré avoir deux cartes différentes. Heureusement, nous avions soudé une carte supplémentaire (car nous avions assez de composants).
Néanmoins cette deuxième ne fonctionne pas alors que nous avons utilisé les mêmes composants que pour la première. Nous avons donc essayé de comprendre le problème de cette carte. Même si les résistances semblent être des résistances 10kΩ, nous avons découvert, au multimètre, que leur valeurs est de 5kΩ. Nous pensons donc à une erreur du constructeur. Cette erreur de valeur est fatale pour la carte car les résistances sont des résistances Pull Up.

➢ Nous n'utilisons finalement pas le gyroscope pour cause de manque de précision. Il aurait fallu le coupler avec accéléromètre et le faire fonctionner par interruption. Mais le timer2 d'interruption de l'arduino ne fonctionne pas avec la communication I2C car ils utilisent la même ressource. A cause de tous ces détails et par manque de temps nous ne permettons pas à notre Robot d'utiliser le gyroscope pour donner la position angulaire.

➢ Nous nous sommes rendu compte de quelques erreurs sur les créations des cartes éléctroniques (plan de masse non présent sur toute la carte, pin non reliées...), nous avons corrigé ces erreurs sur les fichiers de Wiki. Nous ne referons plus ces erreurs dans le futur. Et nous avons dû prononcer d'avantage certaines soudures qui n'étaient pas correctes.

3.2 Résultat final et perspectives

En ayant relu nos objectifs de début de parcours nous pouvons dire que la majeure partie d'entre eux sont atteints. La carte finale n'est malheureusement pas totalement exploitable mais elle reste fonctionnelle.

Nous avons donc, pour la fin de ce projet, créer un Robot mobile. Celui ci est muni d'une pince commandée par un moteur, d'un système de chenille pour le déplacement commandé par des moteurs DC.

Le Robot est simplifié et optimisé au maximum avec un nombre limité de fils et également des supports pour les capteurs et moteur. Il utilise différents capteurs (ultrasons, suiveur de lignes, infrarouge, fins de course).

Beaucoup de pièces sont disponibles dans le Kit pour créer un Robot selon son envie.

Nous avons tenu notre promesse : nous offrons à l'utilisateur l'accès Open Source à des librairies pour faciliter la programmation sous IDE Arduino. Les plus débutants peuvent programmer leur Robot grâce à la programmation en blocs. De plus, le Robot peut être programmé avec une application Bluetooth et de nombreux exemples peuvent aider l'utilisateur.

Néanmoins nous sommes conscients que notre souhait d'obtenir un Kit moins cher que ceux du commerce n'est pas tout à fait à l'ordre du jour. Notre Projet a nécessité beaucoup d'argent. Voici une approximation :

2* Motoréducteur + encodeur	20 euros/moteurs
Moteur DC pince	2 euros
Moteur bras	10 euros
Vis, câbles RJ25, câbles	50 euros
2 Batteries lithium-ion	5 euros/batteries
Capteurs ■ 2 Capteurs de distance par Infrarouges de type SHARP ■ 1 Gyroscope ■ 2 Capteurs de lignes ■ 2 Fin de courses ■ 2 Capteurs de distance Ultrasons	10 euros/sharp 10 euros 4 euros/capteur de ligne 1.5/fin de courses 4 euros/ultrason
Réalisation de cartes à l'extérieur	Prix inconnu
Imprimante 3D et PLA	Imprimante à prix conséquent si nous n'en avons pas à disposition + 10 euros de PLA
Composants pour la carte principale	40 euros
Plexiglas	10 euros
Composants pour les 7 cartes capteurs	18 euros
Composants pour 3 contrôleurs moteurs	10 euros
TOTAL : 249 euros	

Le prix ici correspond à l'ensemble des pièces du Kit dans le cas où nous devions tout acheter. Cependant pour notre Kit nous avons récupéré des matériaux de Robotech ainsi que personnels et M. Redon possédait également quelques matériaux dans la réserve, ce qui a permis de réduire les coûts .

Au départ, nous n'avions pas pensé au prix des composants des cartes lorsque nous avions imaginé notre kit. Le prix s'est avéré assez conséquent. En effet les fournisseurs homologués coûtent assez cher lorsque l'on ne commande pas en gros. Nous avions besoin de plusieurs composants différents en petites quantité. Pour réduire les coûts, certains sites non reconnus par Polytech sont beaucoup plus abordables.

Nous sommes également conscients que l'utilisateur va prendre du temps à réaliser les pièces de son Kit contrairement à l'achat d'un Kit tout fait. Mais faire son propre Kit sera un moyen d'apprendre comment sont fabriquées chaque pièce et chaque carte électronique. Il pourra également fabriquer autant de pièces qu'il le désire en se basant sur nos plans, ou en modélisant de nouvelles pièces. De plus l'utilisateur gagne un temps précieux sur la programmation grâce à nos librairies.

Pour aller plus loin, nous avons pensé à des axes d'amélioration :

- Pour qu'il soit utilisable lors de la coupe de France, il serait intéressant d'ajouter à notre Kit une caméra Pixy car le repérage de couleur est très présent dans les cahiers des charges de la Coupe de France.
- Nous pourrions penser éventuellement à programmer notre Robot en langage C.
- Au départ nous voulions réaliser nous même les capteurs. Mais par manque de temps et surtout par complexité nous avons commandé des capteurs tout faits. Créer les capteurs peut être une amélioration de notre Kit. Mais nous proposons tout de même une solution pour utiliser les capteurs simplement grâce à nos cartes I2C.

Conclusion

Ce projet nous as permis de développer des compétences dans différents domaines comme l'électronique, l'informatique, la mécanique. Nous avons également fait de la gestion de projet : c'est une compétence essentielle à acquérir pour notre futur. Nous nous sommes organisés pour optimiser l'avancée du projet. La gestion du temps a été la plus dure car nous avions beaucoup de petites choses à faire. La conception des cartes et leur débogage a été plus long que prévu. Nous aurions pu faire fonctionner l'entièreté de la carte principale si nous avions eu plus de temps.

Nous sommes fiers du résultat, nous avons réalisé nos objectifs. Notre Robot créé à partir de nos pièces et de nos codes de programmation est fonctionnel.

Annexe 1 : Carte Principale

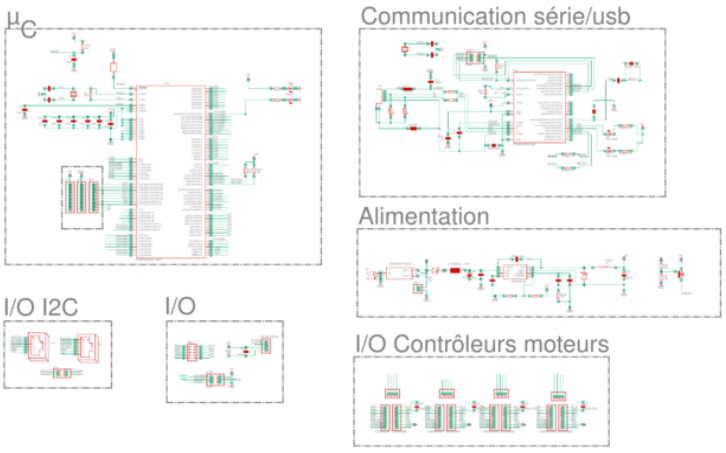


Figure 6 : Schematic de la carte principale sous Eagle

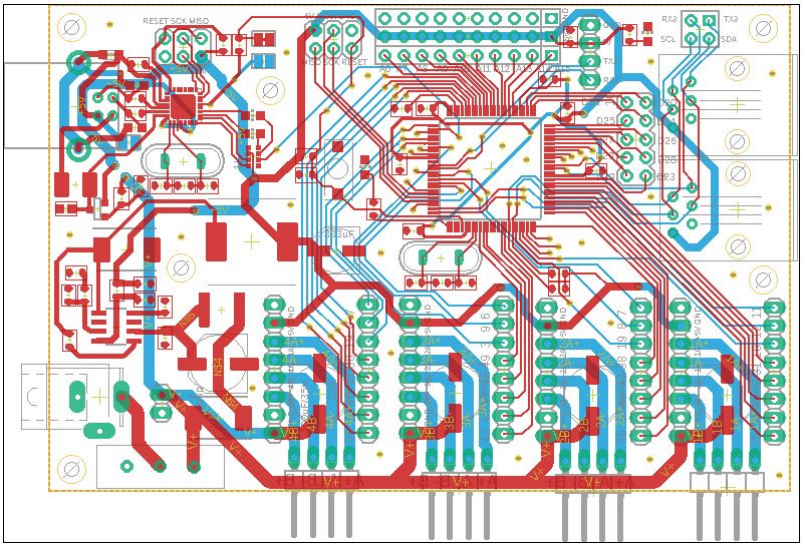


Figure 7 : Routage de la carte principale sous Eagle

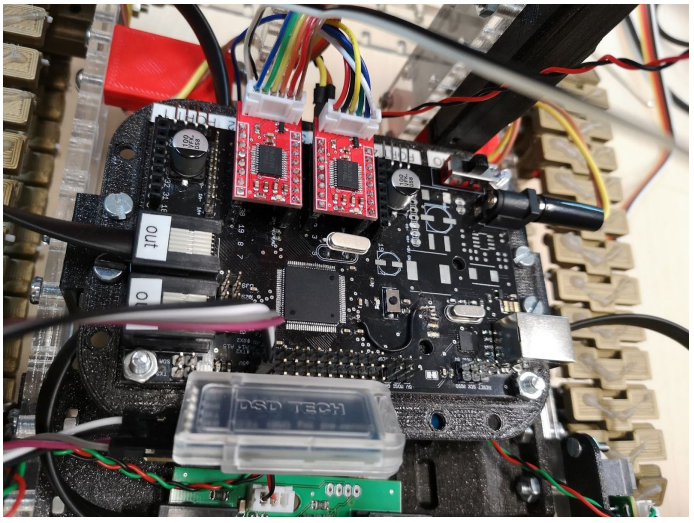


Figure 8 : La carte principale

Annexe 2 : Cartes Capteurs

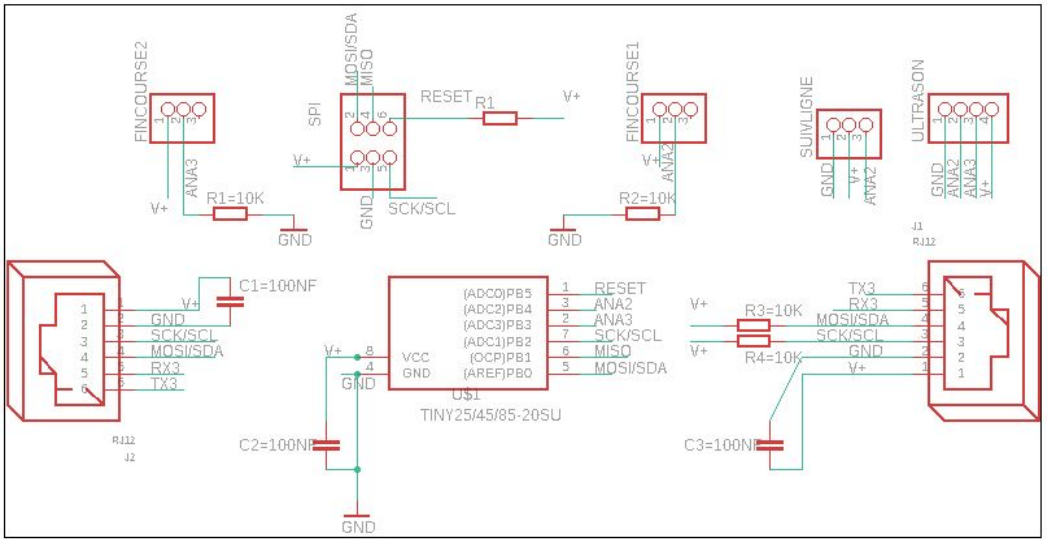


Figure 9 : Schematic de la carte capteur ATtiny85 sous Eagle

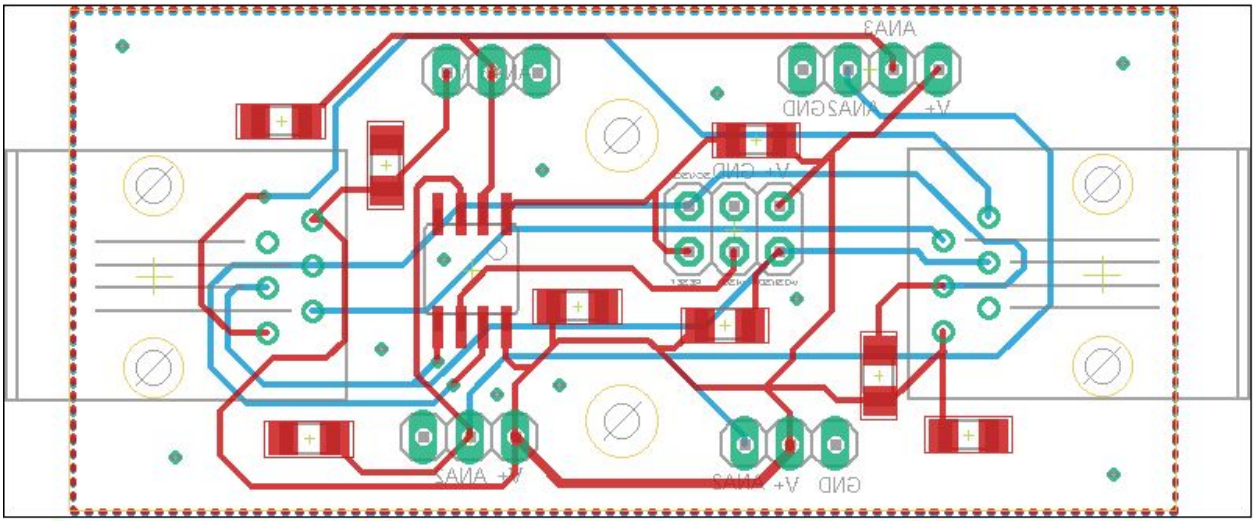


Figure 10 : Routage de la carte capteur ATtiny85 sous Eagle

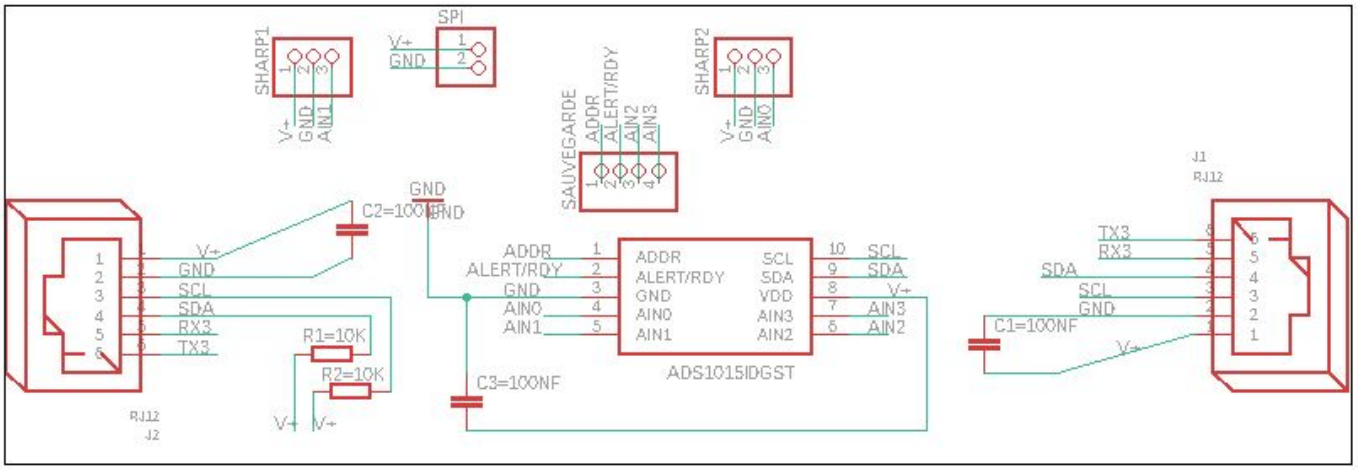


Figure 11 : Schematic de la carte capteur ATtiny85 sous Eagle

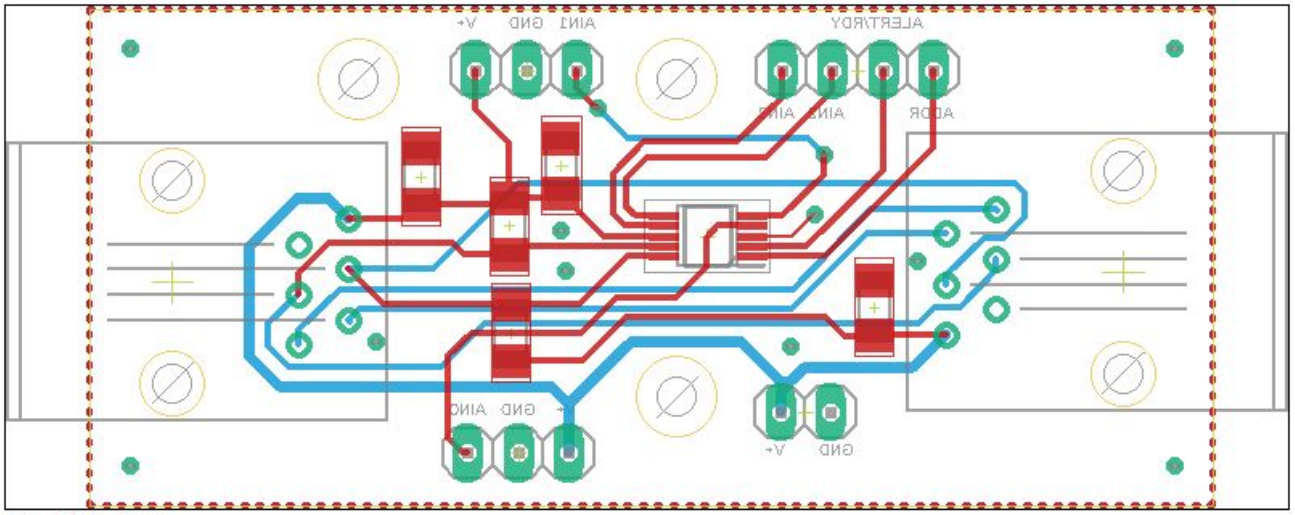


Figure 12 : Routage de la carte capteur SHARP sous Eagle

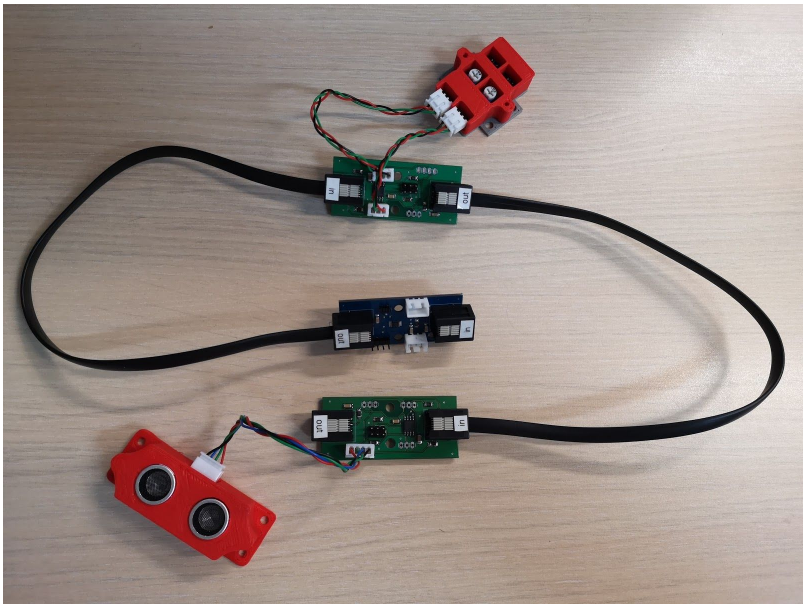
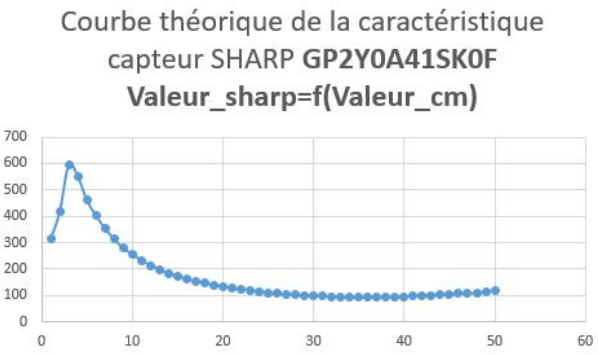


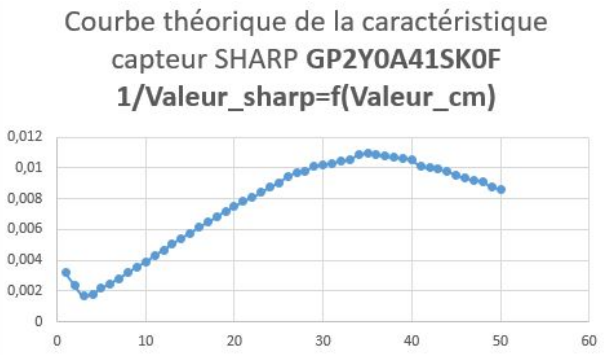
Figure 13 : Cartes capteurs (ATtiny85 en vert et ADC en bleu)

Annexe 3 : Caractérisation Capteur Sharp

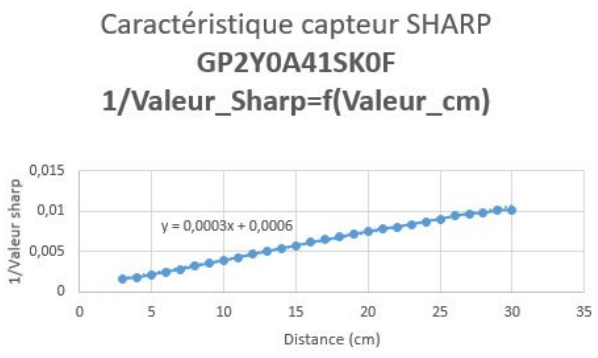
Pour pouvoir utiliser le capteur Sharp nous avons besoin de traduire ce que renvoie le capteur en centimètres.

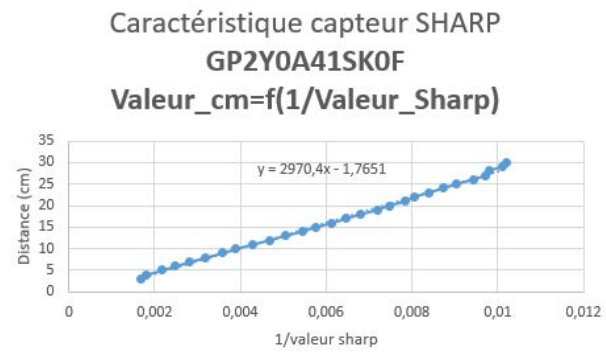
En effet la tension retranscrite par ce capteur n'est pas proportionnelle à la distance. Une courbe caractéristique (Tension en fonction de la distance) est fournie dans la datasheet mais l'équation n'est pas fournie et nous préférons la mesurer par nous même. De plus il existe de nombreux type de capteurs Sharp et même si la forme est semblable, chacun a une courbe caractéristique qui lui est propre. Pour cela nous avons effectué plusieurs mesures en mettant un objet à différentes distances du capteur (de 0cm à 50cm). Le but est d'obtenir une équation traduisant les cm en valeur sharp et inversement.





Pour la courbe Valeur_sharp=f(Valeur_cm), nous savons qu'il sera difficile d'obtenir une équation exploitable. En effet, nous aurons premièrement un problème pour les valeurs de 0 à 3 cm car, au vu de la forme de pic de la courbe, les valeurs pour la capteur sharp seront les mêmes entre 0 et 3cm que pour 3 et 8 cm. Nous ne pourrons donc pas être précis. Nous avons également un problème pour les valeurs du capteur Sharp après 33cm, elles sont quasiment toutes pareil. Cette courbe ne nous permet donc pas d'associer une valeur du capteur à une valeur en cm.
Nous avons donc fait un test avec la courbe 1/Valeur_sharp=f(Valeur_cm), nous observons qu'entre 3 et 33 cm la courbe est linéaire. Cela permet d'obtenir une équation pour traduire les valeurs en cm avec celles du capteur Sharp.
Nous avons donc refait les courbes en 1/Valeur Sharp entre 0 et 33cm pour obtenir l'équation de passage de cm à valeur Sharp et inversement.





Nous avons donc admis les équations suivantes :
1/Valeur_sharp = 0,0003*Valeur_cm + 0,0006
Valeur_cm = 2970,4*(1/Valeur_sharp) - 1,7651
Notre capteur est donc précis entre 3 et 33cm.

Annexe 4 : Cartes Contrôleurs Moteurs

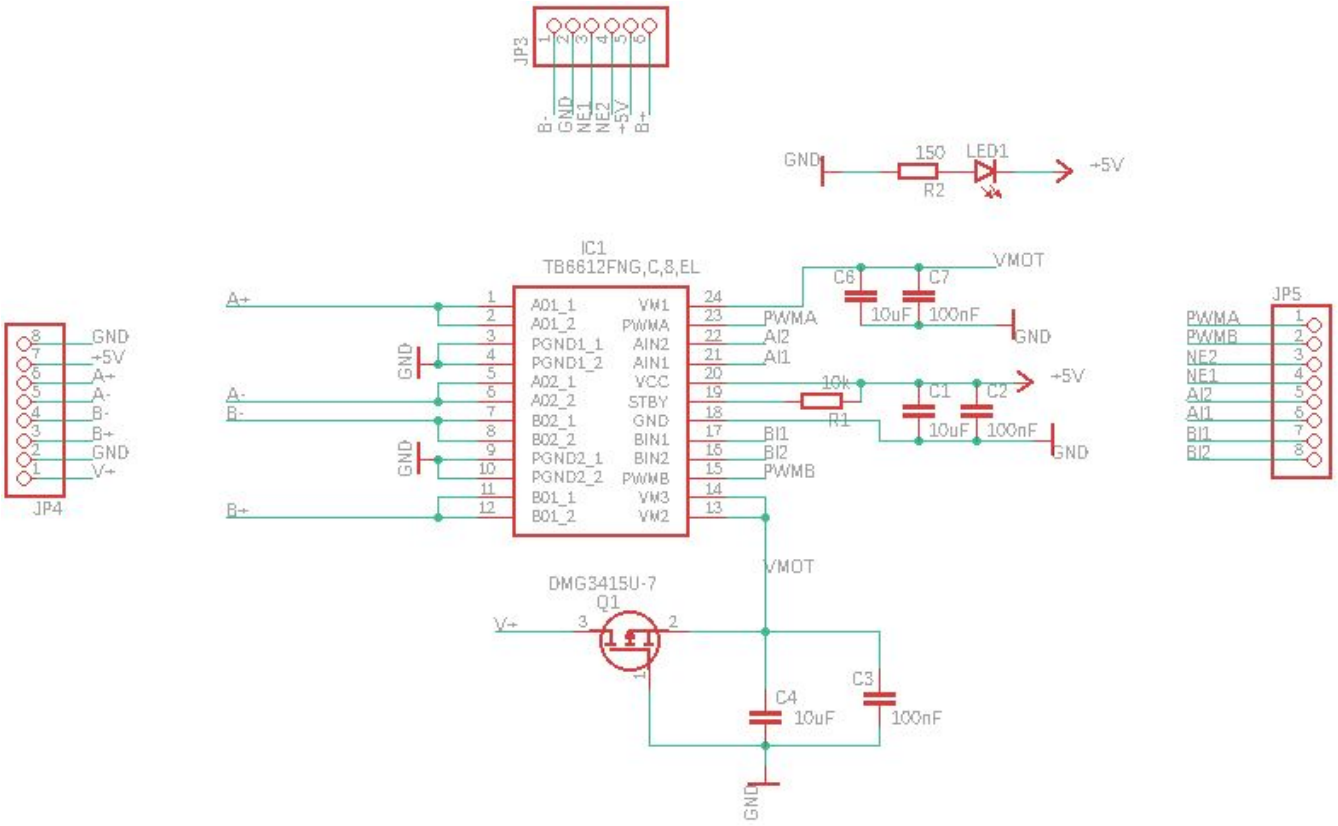


Figure 14 : Schematic de la carte contrôleur moteur sous Eagle

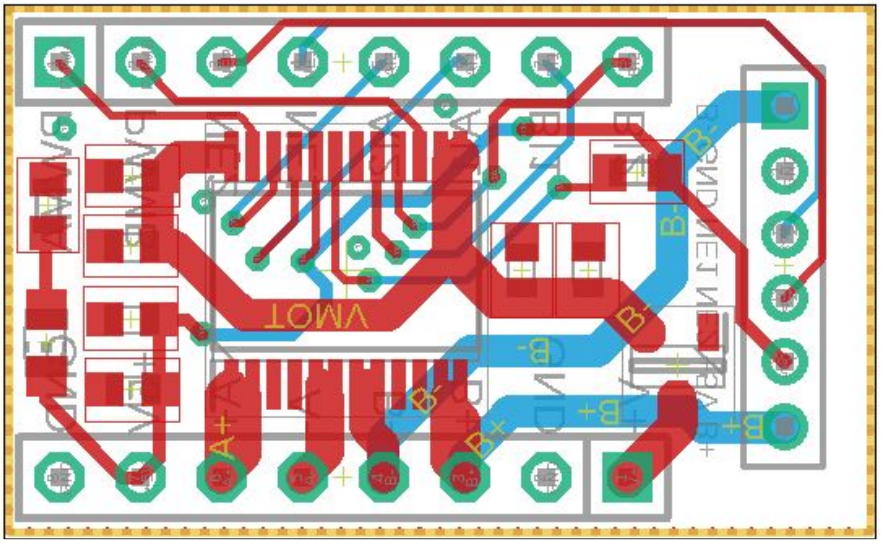


Figure 15 : Routage de la carte contrôleur moteur sous Eagle

Annexe 5 : Programmation

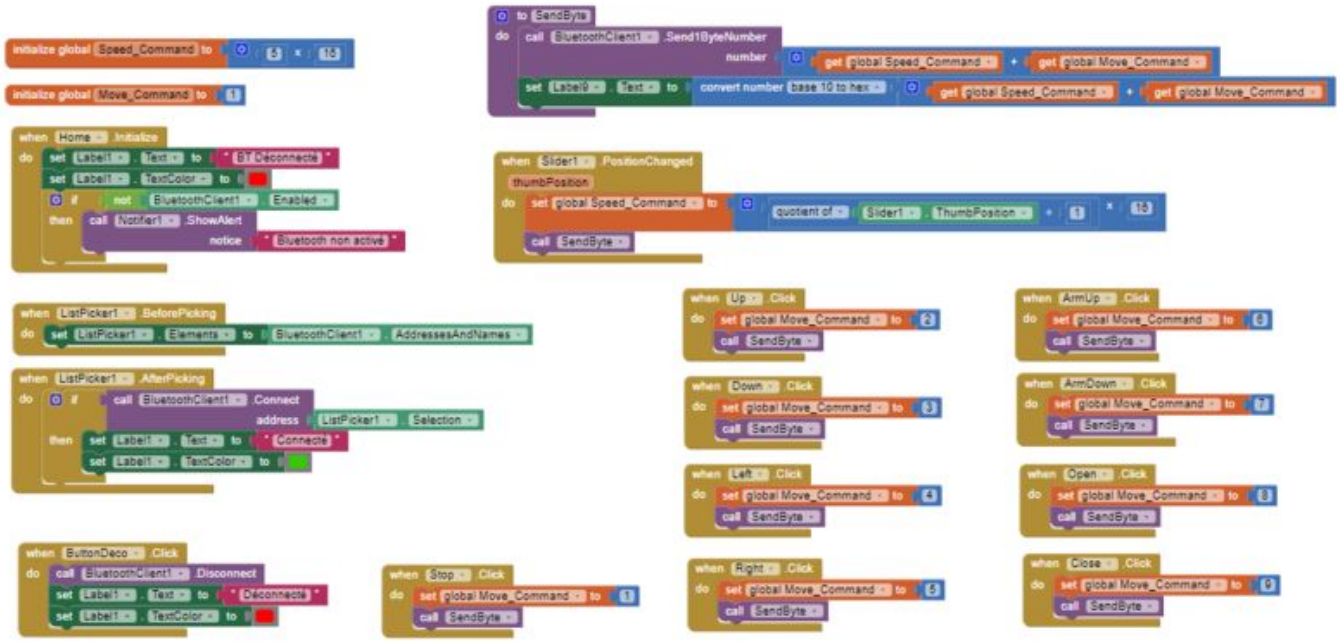


Figure 16 : Code de l'Application

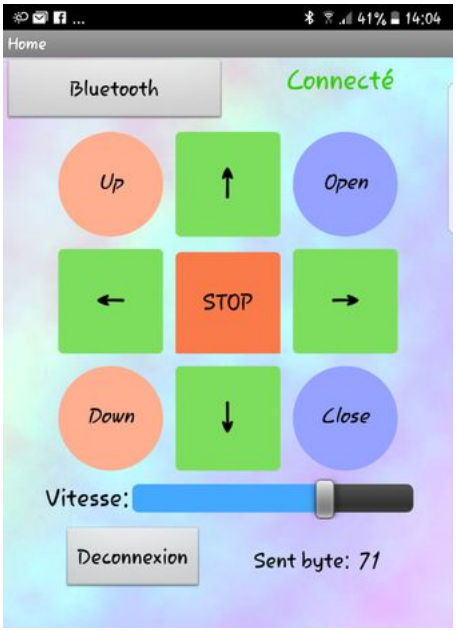


Figure 17 : Visuel de l'application et QRCode pour le téléchargement

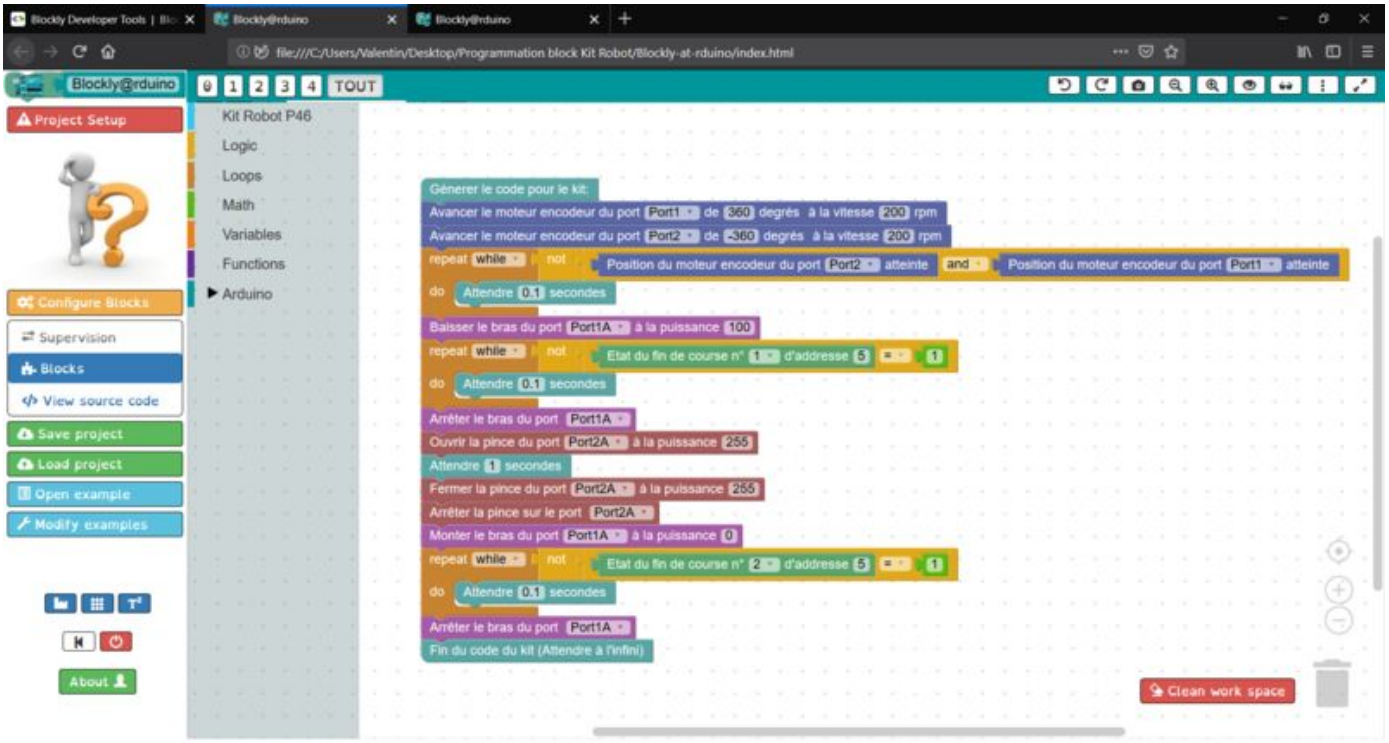
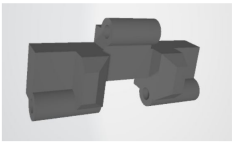


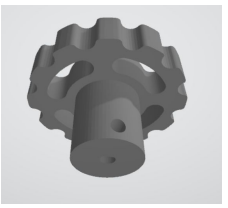
Figure 18 : Blockly Arduino avec nos libraires

Annexe 6 : Catalogue des Pièces du Kit

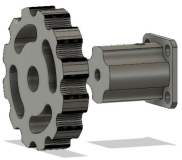
Maillon Chenille



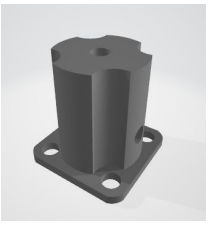
Roue moteur chenille



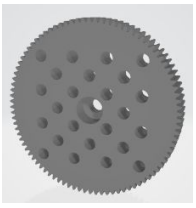
Roue libre chenille et support



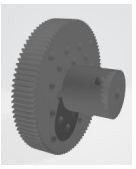
Support roue libre



Roue libre sans chenille

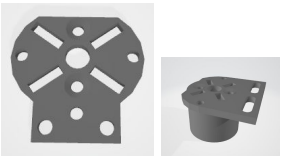


Roue moteur sans chenille

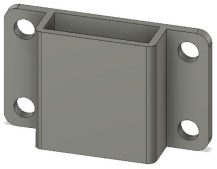


Renforcement Moteur

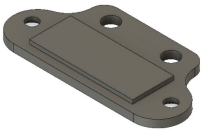
Support Moteur



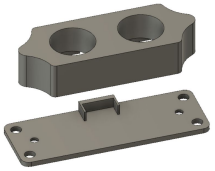
Support capteur fin de course



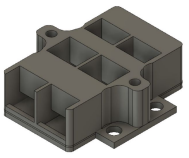
Support capteur infrarouge (Sharp)



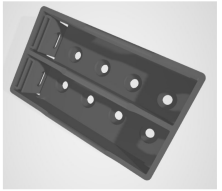
Support capteur Ultrason



Support suiveur de ligne



Support Batteries



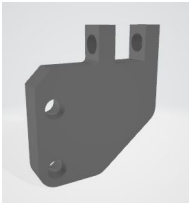
Support Carte Principale

Chargeur de Batterie

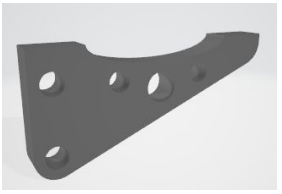
Moyeu et Tige

Support Bras Pince

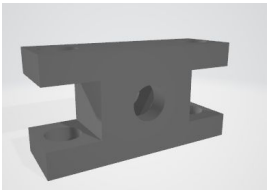
Base de la Pince



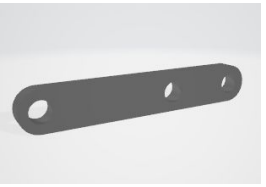
Crochet de la pince



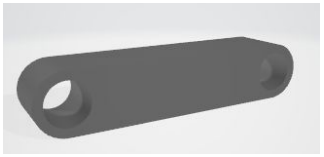
Ecrou central de la pince



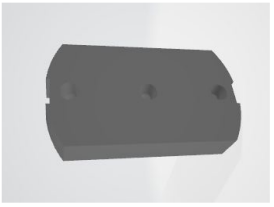
Grande liaison de la pince



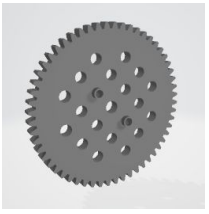
Petite liaison de la pince



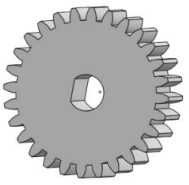
Liaison de la pince



Grand Engrenage moteur Bras

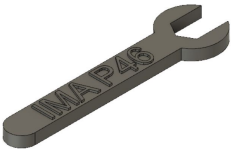


Petit Engrenage moteur Bras



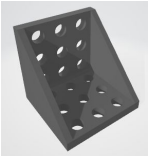
Support Gros Moteur DC pour le Bras

Clé Anglaise



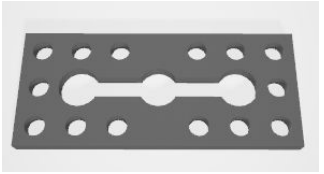
Boîte à Vis

Coin 3*3

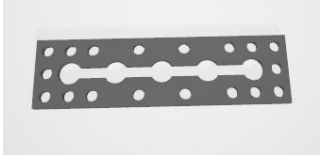


Angle 45° 12 trous

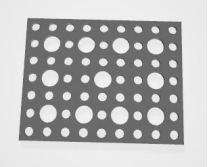
Petite Bande



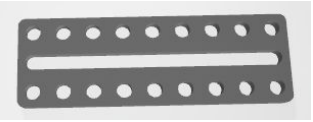
Grande Bande



Grande pièce



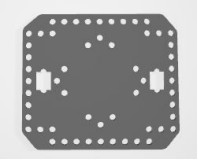
Pièce Longue



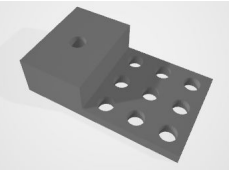
Coin 2*4

Coin 3*3

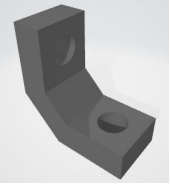
Base



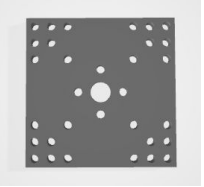
Pièce centre 3*3



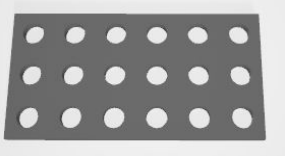
Petite pièce Coin



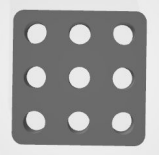
Grande pièce avec trou central



Pièce plate 6*3



Pièce plate 3*3

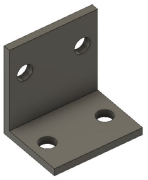


Pièce 6 trous

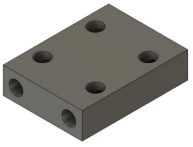
Pièce courbée 6*2

Pièce ½ courbée

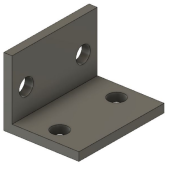
Angle 4 trous long



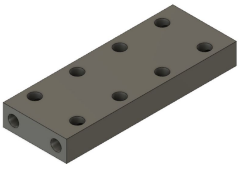
Pièce épaisse 2*2

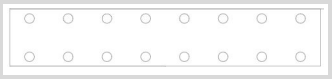
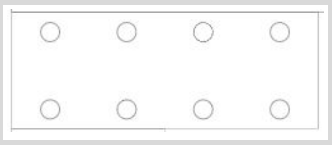
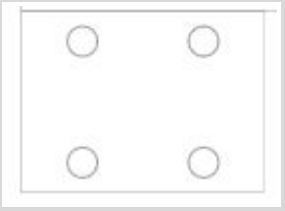


Angle 4 trous court



Pièce épaisse 4*2



Large 192*24mm, 2*12 trous		Fine 188*8mm, 12 trous	
Large 176*24mm, 2*11 trous		Fine 140*8mm, 9 trous	
Large 128*24mm, 2*8 trous		Fine 92*8mm, 6 trous	
Large 64*24mm, 2*4 trous		Fine 76*8mm, 5 trous	
Large 32*24mm, 2*2 trous			

Annexe 7 : Evolution du Robot

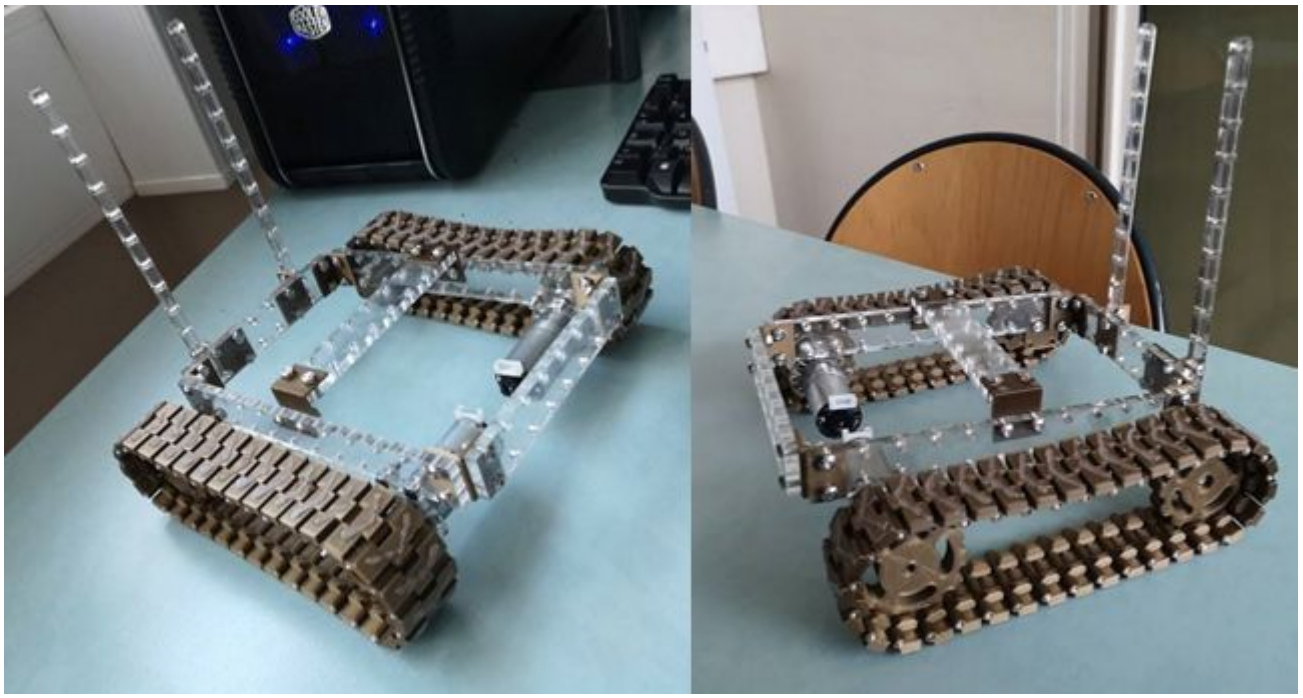


Figure 19 : Châssis du Robot

Figure 20 : Châssis du Robot avec pince

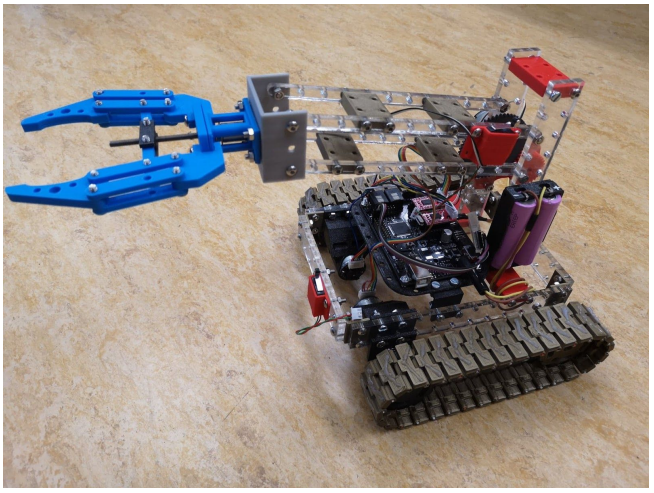


Figure 21 : Robot avec les moteurs et capteurs