



Projet Tutoré

***ROBOT D'EXPLORATION EN MILIEU
HOSTILE***

***BERLAND BASTIEN, DAVILLARS GREGORY,
THERON JOSHUA***

Enseignant référent : Joaquin Jorge Rodriguez

Table des matières

Contexte	4
Qu'est-ce que la robotique ?	4
Quels sont les différents types de robot ?	4
Préambule du projet	5
Cahier des charges.....	6
Contexte du projet	6
Périmètre du projet.....	6
Aspects fonctionnels et techniques et analyse des besoins.....	6
Ressources.....	7
Histoire	7
Chaîne fonctionnelle et diagrammes SySML.....	8
Gantt	12
Choix des composants	13
Nomenclature	14
WIFIBOT	15
Description SUB-D9 mâle :.....	16
Commande du WiFiBot avec ROS	19
Modélisation 3D	20
Manipulateur 5 axes	Error! Bookmark not defined.
Commande du manipulateur avec Ros	Error! Bookmark not defined.
Montage électronique	29
Partie programmation vision	30
Mise en commun et assemblage	34
Quelles optimisations possibles ?	35
Difficultés :	35
Avis personnel	36
Remerciements.....	37
Sources.....	38

Projet tutoré 2023-2024

Annexes	39
Plan de définition	39
Schéma U	48
Programme Python.....	49

Contexte

Qu'est-ce que la robotique ?

La robotique est un domaine d'activité recouvrant l'étude, la conception et la fabrication de robots ou machines automatisées. Elle implique des compétences en informatique, en électronique et en mécanique. De ce fait, pour être considérée comme un robot, une machine doit être pourvue de capteurs qui analysent l'environnement, mais également d'un système de traitement logique et d'actionneurs.

A partir des informations obtenues au sein de son environnement, le robot peut agir de manière autonome et intelligente en se basant sur des modèles d'apprentissage embarqués ou du Machine Learning.

Quels sont les différents types de robot ?

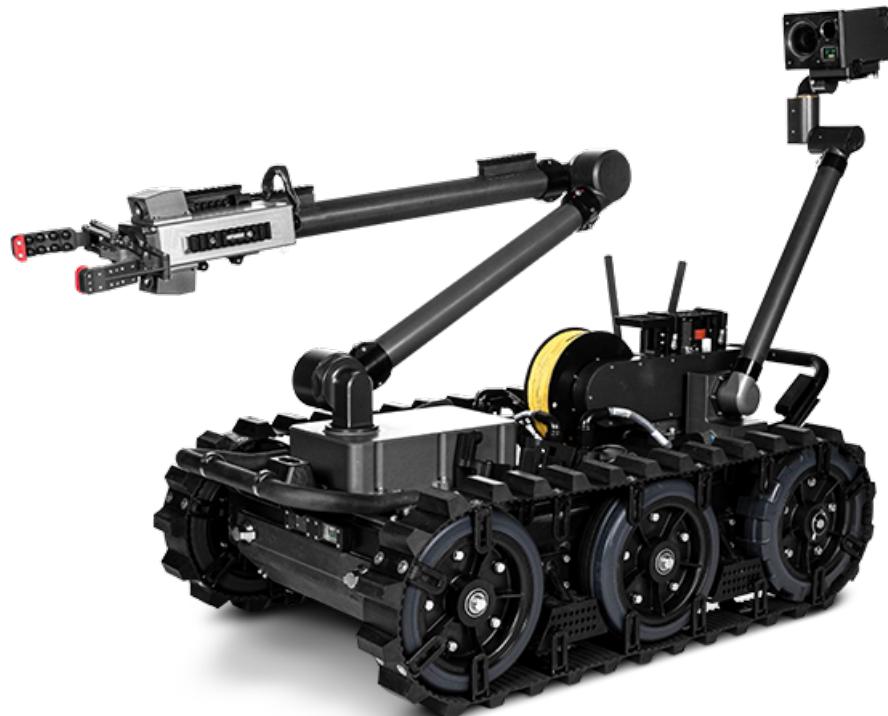
La robotique englobe un champ d'activité très large et diversifié. Dont voici les principaux et leurs champs d'application avec quelques exemples existants :

- ② **INDUSTRIE** : robot de gestion de chaîne d'assemblage
- ② **ARMEE** : drone, robot-espion, robot-mule
- ② **SECURITE** : vidéosurveillance
- ② **SANTE** : échographie, chirurgie assistée
- ② **AEROSPATIAL** : robot explorateur de la NASA
- ② **TRANSPORT** : voiture autonome
- ② **USAGE DOMESTIQUE** : robot aspirateur, robot tondeuse
- ② **ACCOMPAGNEMENT** : jouet automatisé, robot humanoïde
- ② **INFORMATIQUE** : chatbot, assistant vocal

Préambule du projet

Notre groupe de trois étudiants en 5ème année a cette année choisi pour projet d'étude la réalisation d'un robot d'exploration en milieu hostile. Un sujet complet qui nous permettra de renforcer nos compétences en mécanique, électronique et programmation.

Nous avons souhaité nous orienter sur un robot destiné à être utile dans de nombreux domaines. Notre robot d'exploration en milieu hostile pourrait être destiné à l'industrie, à l'armée ou à la sécurité, s'adaptant au besoin en changeant le préhenseur de son bras (robot manipulateur, robot démineur, etc.). Leur précision permet de limiter les risques.



Cahier des charges

Contexte du projet

Aujourd’hui, un nombre significatif d’interventions sont sujettes à de gros risques où une intervention humaine est nécessaire (guerre, accidents nucléaire, fuite de produit chimique, zones accidentées, etc.). C’est pourquoi, dans le cadre de notre projet robotique tutoré, nous avons souhaité nous orienter sur un robot destiné à être utile à de multiples fins. Notre robot d’exploration en milieu hostile pourrait être utile dans de nombreux domaines tels que l’industrie, l’armée ou la sécurité, s’adaptant au besoin en changeant le préhenseur de son bras (robot manipulateur, robot démineur, etc.). Sa précision et son contrôle à distance permet de limiter au maximum les risques.

Les objectifs du projet sont les suivants :

- ✓ Exploration de milieu à risques
- ✓ Limiter les dégâts matériels
- ✓ Remplacer l’humain pour des explorations à risques

Périmètre du projet

- ☒ Le robot sera limité à l’exploration terrestre non humide, l’isolation à des dégâts liquides n’est ici pas prise en compte
- ☒ Le projet sera limité au matériel disponible au plateau robotique et à l’iut ainsi que la possibilité d’imprimer des éléments en 3D
- ☒ Il devra être réalisé avant la date de rendu le 15 décembre

Aspects fonctionnels et techniques et analyse des besoins

Dans le but d’explorer des milieux à risques, le robot aura besoin de pouvoir se déplacer sur plusieurs axes. Son déplacement devra être adapté à plusieurs terrains (rocailleux, plat, sableux...), que nous limiterons ici à humide. Afin de pouvoir visualiser à distance ce que filme le robot, il sera équipé d’une caméra intégrée avec vision nocturne pour un renvoi d’image en direct, ainsi que d’un capteur LiDAR permettant la création d’une cartographie. Cette caméra devra être capable de détecter l’objet souhaité à distance. Il aura aussi besoin d’un bras robotisé pour interagir avec l’extérieur commandable et automatisé.

Nos besoins matériels sont les suivant :

- Un châssis avec sa batterie
- Un moteur brushless pour se déplacer
- Une caméra
- Un bras robotisé

Ressources

Ressources techniques

Les ressources mise à notre disposition sont l'équipement robotique du plateau, L'accessibilité aux imprimantes 3D, un fer à souder ainsi que tout l'outillage de base.

Ressources intellectuelles

Nous remercions tout le corps enseignant ainsi que les chercheurs du plateau qui ont bien voulu répondre à nos questions.

Délais

Le projet devra être réaliser entre les dates du 4 septembre 2023 et du 15 décembre 2024.

Histoire

L'origine de ce projet remonte au sud de la France, où un groupe d'amis s'adonne depuis plusieurs années à l'exploration des nombreux affluents des rivières environnantes. Un jour, l'un d'entre eux découvrit l'entrée d'une grotte, suscitant ainsi la curiosité des jeunes aventuriers. Malheureusement, la grotte se révéla bien trop dangereuse pour une exploration humaine approfondie. Face à cette réalité, l'idée d'utiliser un robot télécommandé pour explorer ce tunnel accidenté émergea. Cependant, étant donné le coût élevé de ces robots sur le marché, la construction apparut comme la seule option viable. C'est ainsi que naquit l'idée de concevoir un robot d'exploration dans le but de parcourir des cavernes et d'autres lieux ensevelis.

Chaîne fonctionnelle et diagrammes SySML

➤ Chaîne fonctionnelle :

La chaîne fonctionnelle est utilisée dans la conception et surtout dans l'amélioration d'un système.

- La chaîne d'information :

La chaîne d'information pilote la chaîne d'énergie et comprend les trois fonctions techniques suivantes : acquérir, traiter et communiquer.

Les informations de nature logique ou analogique, circulent à travers le système afin d'obtenir des ordres d'exécution et des comptes rendus à destination des utilisateurs ou d'autres systèmes.

- La chaîne d'énergie :

La chaîne d'énergie comprend les quatre fonctions techniques suivantes : alimenter/stocker, distribuer, convertir et transmettre. Elle reçoit des ordres venant de la chaîne d'information.

L'énergie peut prendre plusieurs formes (mécanique, électrique, thermique, chimique...). Elle circule à travers le système afin de réaliser des actions sur la matière d'œuvre pour apporter la valeur ajoutée.

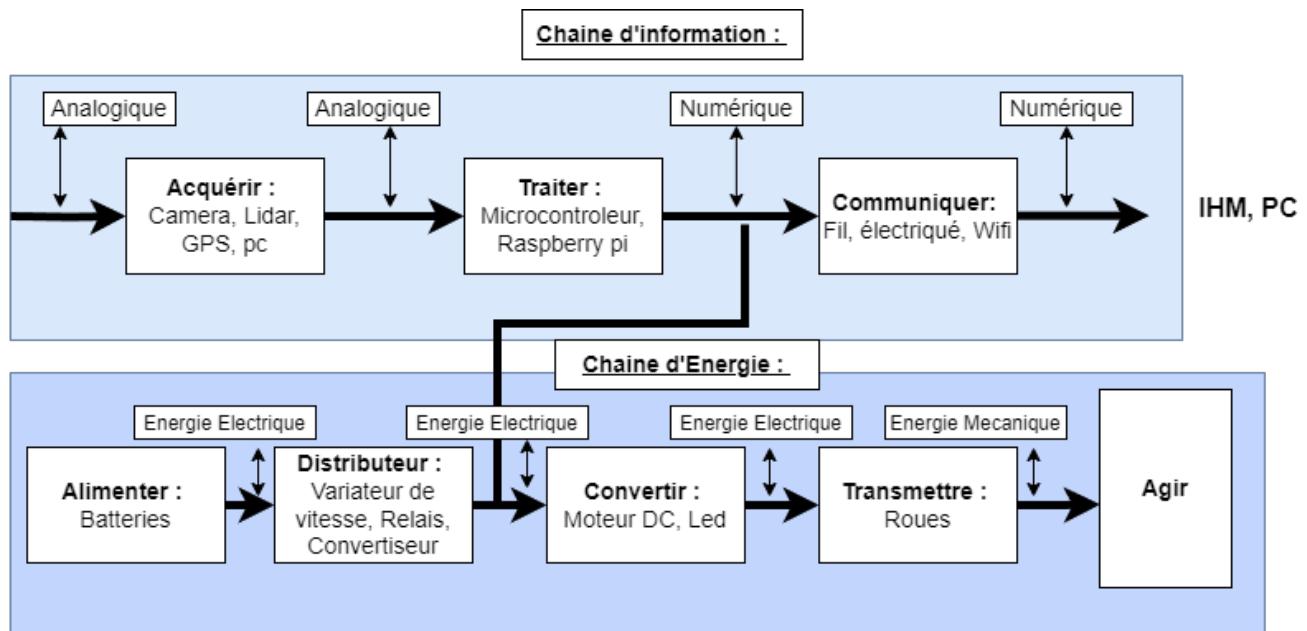


Diagramme des exigences :

C'est un diagramme fonctionnel. Il décrit les exigences du cahier des charges fonctionnel. Une exigence exprime une capacité ou une contrainte à satisfaire par un système. Elle peut exprimer une fonction que devra réaliser le système ou une condition de performance technique, physique, de sécurité, de fiabilité, d'ergonomie ou d'esthétisme.

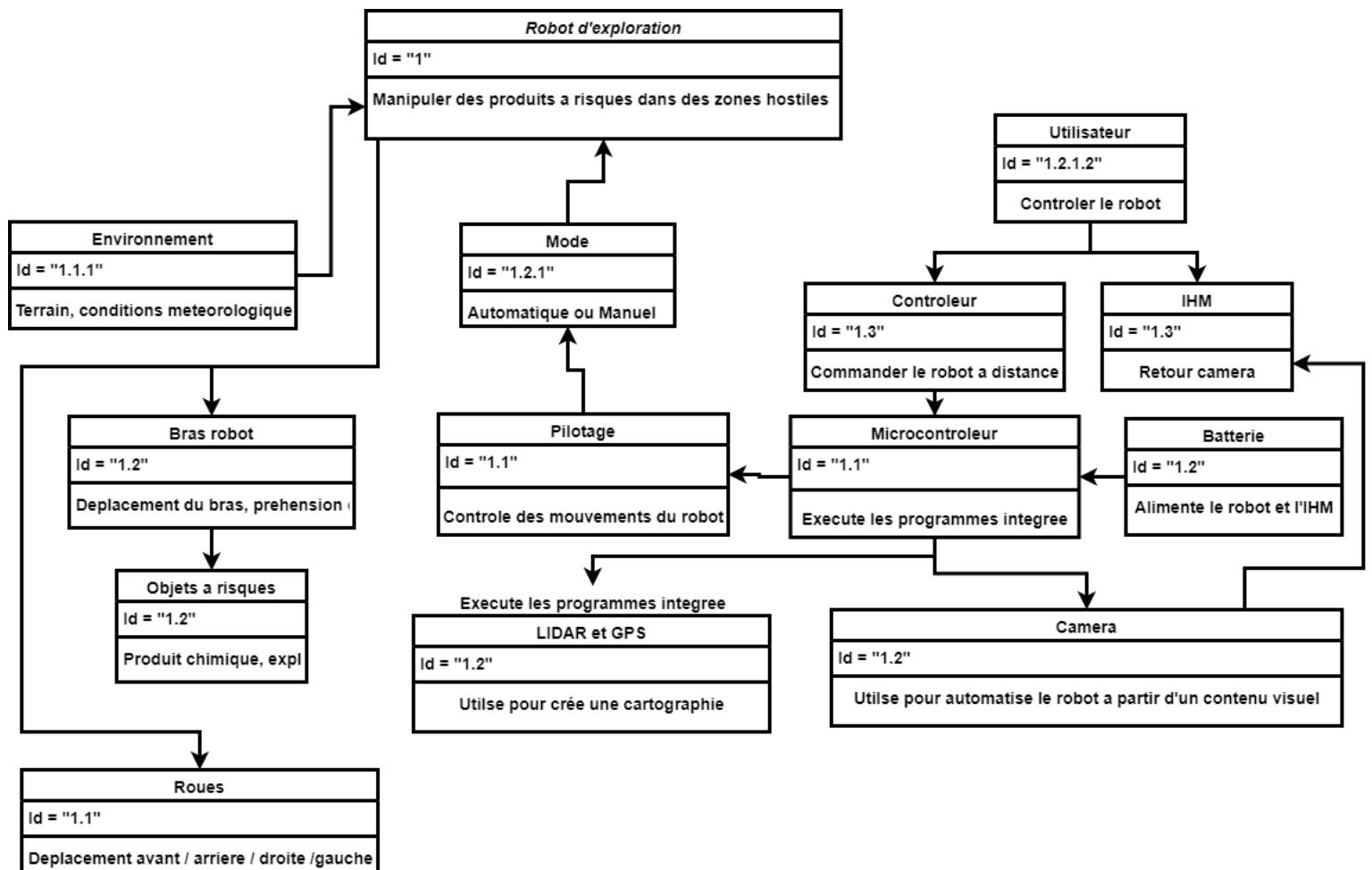


Diagramme des cas d'utilisation :

C'est un diagramme fonctionnel. Il montre les interactions fonctionnelles des acteurs et du système d'étude. Il délimite précisément le système, décrit ce qu'il fera sans spécifier comment (et non ce que fera l'utilisateur). Il exprime les services offerts par le système aux utilisateurs (Actor).

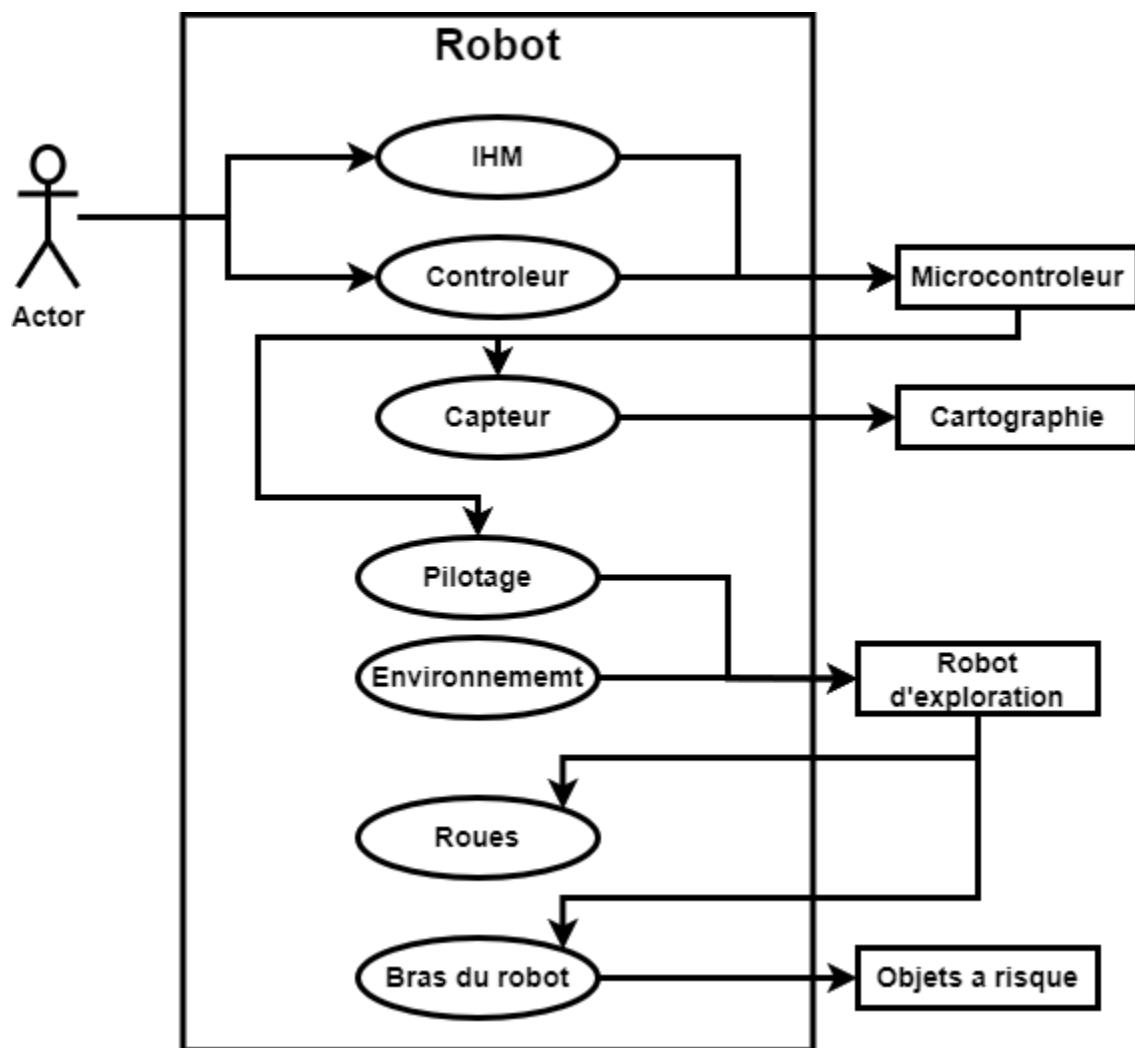
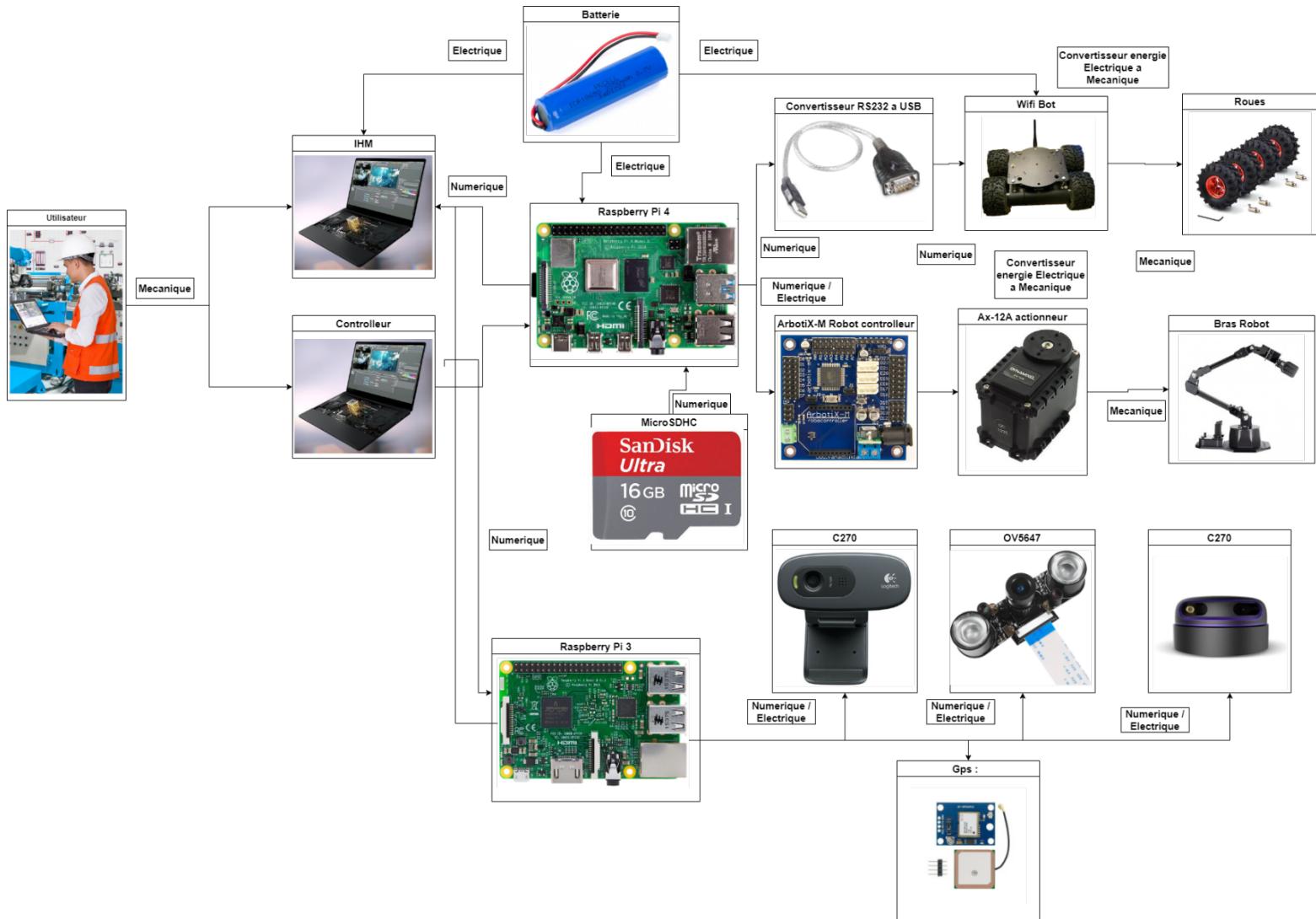


Diagramme de définition de blocs :

C'est un diagramme statique. Il montre les briques statiques : blocs, composition, associations... Il est utilisé pour décrire l'architecture matérielle du système.

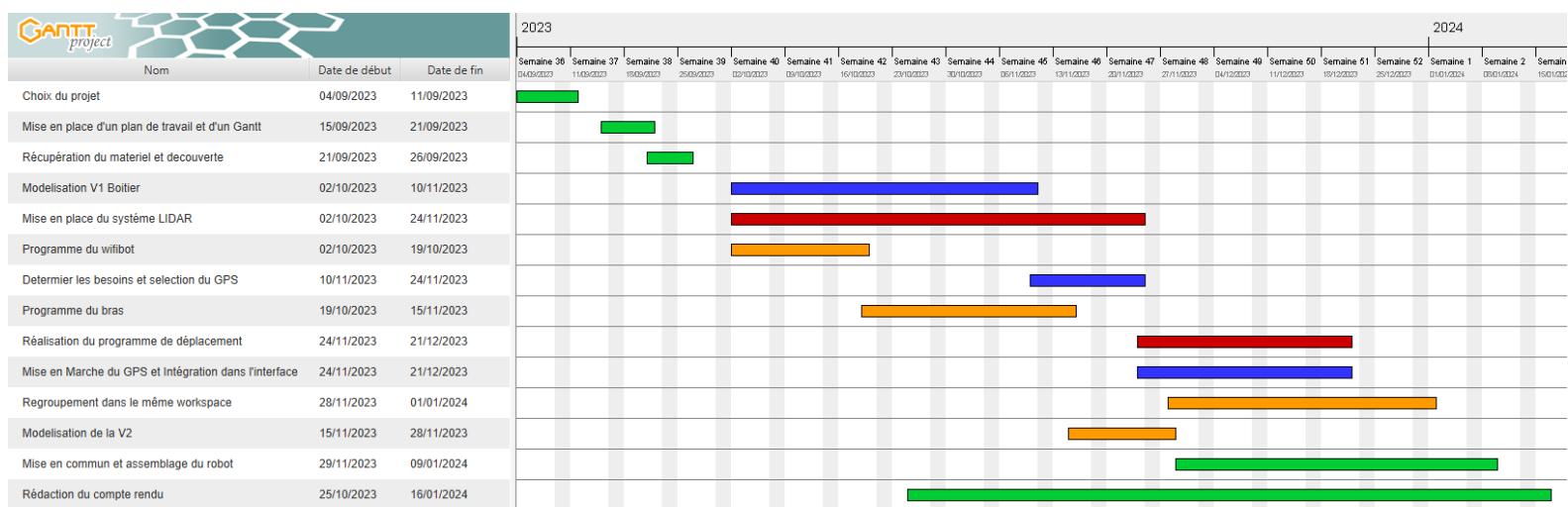
Un bloc est une entité bien délimitée qui encapsule principalement des attributs (variables d'état), des opérations (procédures comportementales), des contraintes, des ports (échange de flux avec l'extérieur) et des parts (sous-blocs internes).

Un bloc peut modéliser tout le système, un élément matériel ou logiciel.



Gantt

C'est l'outil de planification par excellence. Il permet de planifier notre projet de la meilleure façon possible et de visualiser rapidement l'avancement des différentes tâches à réaliser. C'est aussi un bon outil de communication puisqu'il a permis à tous les membres de l'équipe de suivre le planning établi et de connaître l'avancée du projet en temps réel.



Choix des composants

Le robot est équipé d'un Raspberry Pi 4 et un Raspberry Pi 3, essentielle pour nous permettre de centraliser la commande des autres composants puisqu'elle fait office ici de carte mère embarquée. Grâce à ses ports USB, la Rasp gère la connexion et le retour d'informations entre l'utilisateur, la Webcam et le WiFiBot. De plus, la mise en place d'un réseau wifi nous permet la commande à distance du robot. Muni d'une carte SD de 16 Gigas, nous avons choisi d'utiliser l'OS Raspbian mais il serait tout à fait convenable d'utiliser n'importe quel autre OS.

Le WiFiBot Robotics est une plateforme robotique Open Source. Il est divisé en deux parties distinctes, la plateforme mobile et la carte de commande. La partie plateforme mobile est composée d'un châssis en acier, d'une batterie, de quatre moteurs et de quatre contrôleurs Brushless ainsi que d'une carte électronique.

Depuis cette carte de commande, nous obtenons beaucoup d'informations telles que le niveau et l'utilisation de la batterie, l'état de la communication avec la carte de commande et les retours des contrôleurs. L'ArbotiX-M Robot Controller, lié au bras, que nous avons utilisé est un contrôleur de robot polyvalent, c'est une solution de contrôle avancée pour les actionneurs basés sur Bioloid et Dynamixel. En tant que microcontrôleur compatible Raspberry, la carte contrôleur ArbotiX bénéficie d'une énorme communauté open source, de nombreuses bibliothèques et d'exemples. Très utile pour nous et notre projet.

Le bras robotique en lui-même est équipé de servomoteurs AX-12A. Ils nous donnent la possibilité de suivre leur vitesse, leur température effective, leur tension ainsi que leur charge. L'algorithme de contrôle dont nous nous servons maintient la position de l'arbre en ajustant individuellement la position de chaque servos, permettant de contrôler la vitesse et la force de la réponse de l'ensemble du bras. Toute la gestion des capteurs et le contrôle de position sont gérés par le microcontrôleur intégré, ceci est donc parfaitement adapté à notre utilisation que nous voulons centralisée sur la carte embarquée.

Pour notre retour visuel robot-IHM, nous avons opté pour la webcam fournie, la Logitech C270 HD, offrant une résolution de 720p et une fréquence de 30 ips. Cette caméra, associée à un programme Python, nous permet d'obtenir la vision du robot sur l'IHM avec une estimation de la distance robot-objet ciblé. De plus, nous disposons d'une caméra Night Vision OV5647 équipée de deux lumières infrarouges pour les déplacements nocturnes.

Pour permettre une géolocalisation sur le site d'exploration, nous utilisons un module GPS Gy-NEO6MV2.

Afin de créer une cartographie, nous utilisons un capteur LiDAR : le RPLIDAR A3.

Afin d'alimenter proprement la Raspberry pi ainsi que le bras articulé, nous avons besoin de réadapter la tension de sortie de la batterie. Pour cela nous utilisons un convertisseur DC-DC de la marque HiLetgo. Il permet de choisir une valeur de sortie de 1,23 V à 30V.

Nomenclature

<u>NATURE-NOM</u>	<u>NBR</u>	<u>DIMENSION</u> <u>LXWXH</u> <u>(MM)</u>	<u>VALEUR</u> <u>NOMINALE DE</u> <u>RÉFÉRENCE</u>	<u>TENSIONS</u>	<u>MARQUES</u>	<u>PRIX</u> <u>(€)</u>
RASPBERRY-PI 4	1	85 x 56 x 17	64-Bit	5.5V et 3A	Fondation Raspberry Pi	60
RASPBERRY-PI 3	1	85 x 56 x 17	64-Bit	5.5V et 3A	Fondation Raspberry Pi	35
WIFIBOT	1	320 x 370 x 150	Brushless powerful motors	4V x 12V	Nexter Robotics	2000
AX-12A ACTUATOR	5	32 x 50 x 40	Dynamixel servomotors	2V x 1.5A	Robotis	59.92
ARBOTIX-M ROBOCONTROLLER	1	61 x 61 x 10	10 more general purpose I/O pins	12V	Dynamixel	40
C270	1	72x31x66	HD Webcam	5V x 150 mA	Logitech	20
MICROSDHC CONVERTISEUR	1	9x8	16 Giga	x	SanDisk	10
	3	10.6 x 8.6 x 3.5	dc -dc	1.5 V à 35 V	Hiletgo	13
RPLIDAR A3	1	80 x 80 x 45	10 - 25 m	5v x a3	Slamectec	800
OV5647	1	25 x 24 x 3.5	5 MP	5v x a3	WAVESHARE	25
GY-NEO6MV2	1	25x35	9600bauds/s	5v	ICQUANZX	10

WIFIBOT

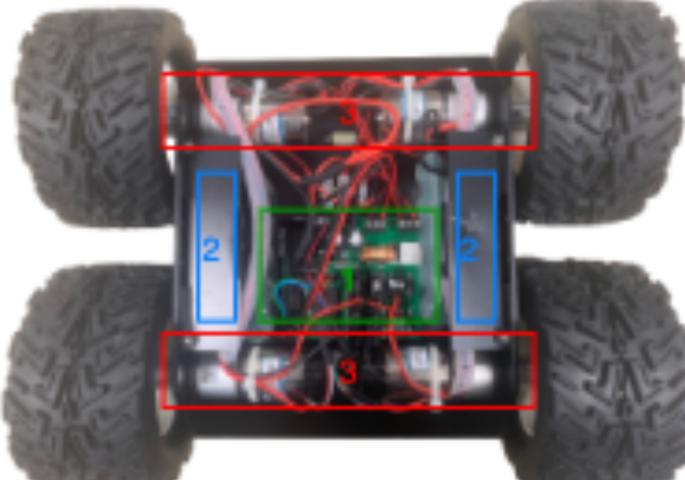
Le wifi bot est un robot d'exploration open source pour la recherche ou le développement. Il est fabriqué par l'entreprise de robotique française Nexter Robotics qui le propose à un prix qui varie en fonction des options de 2000 à 3000 euros.

Il est divisé en deux parties distinctes : la plateforme mobile et la carte de commande.

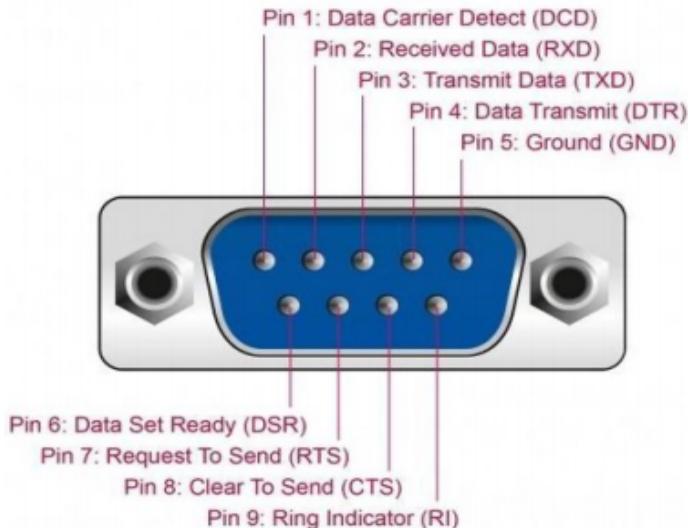
La carte de commande peut varier en fonction des modèles mais les plus courantes sont les Raspberry Pi 3 et 4, le module Jetson Nano/TX2 ou bien encore Xavier AGX/NX. Néanmoins, cette carte de commande est permissive et adaptable car elle permet l'ajout de toutes sortes de modules extérieurs, comme des caméras, des micros, une antenne wifi ou bien une grande variété de capteurs.

De son côté, la partie plateforme mobile est-elle composée :

- D'un châssis en acier avec un dimension de L : 32cm, l : 37 cm, : 15 cm pour un poids de 3.8 kilos
- D'une batterie de 12.8V LIFEPO4 avec une capacité de 10AH.
- De quatre moteurs Brushless de 12v, un train épicycloïdal de 26:1 et 156 rpm
- D'un encodeur de 336 signaux par tour.
- D'une carte électronique permettant la gestion de la batterie, des capteurs, des contrôleurs et des moteurs.
- D'un convertisseur USB/RS232 équipé sur la carte électronique.

- 
1. Carte électronique
2. Batterie
3. Moteur

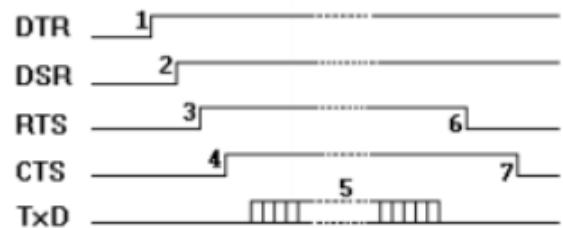
Le protocole RS232 est un protocole de communication standardisé permettant la communication entre deux appareils informatiques. Plus souvent appelé port série ou Port Com. Daté de 1962, standardisé par l'EIA (Electronic Industries Alliance), est un protocole en voie de disparition depuis l'apparition de l'Usb. Le connecteur le plus connu est le SUB-D 9, qui communique entre 75 à 128000 Baudes (bits par seconde). Dans notre cas, le baudrate idéal est situé à 19200, ce protocole nous est donc largement suffisant.



Description SUB-D9 mâle :

N° broche	Fonction et orientations des signaux. (Cible, Envoyeur)		
1	DCD	<---	Déetecte le départ de la transmission
2	RxD (ou RD)	<---	Récupéré les données envoyées par la cible
3	TxD (ou TD)	---->	Envie des données par l'envoyeur
4	DTR/	---->	Confirmation de l'état du terminal/ordinateur
5	SG, GND	----	La masse (GND)
6	DSR/	<---	Données prêtes à l'envoi
7	RTS/	---->	Demande de l'envoi
8	CTS/	<---	Prêt à l'envoi
9	RI	<---	Sonnerie

- 1 DTR L'ordinateur d'envoi indique qu'il peut envoyer
- 2 DSR Il indique que les données sont prêtes à être envoyé
- 3 RTS Il va demander l'autorisation à la cible si il peut émettre
- 4 CTS La cible informe qu'il est prêt
- 5 TxD L'envoie des données par l'envoyeur
- 6 RTS L'envoyeur annule sa demande démission
- 7 CTS L'envoyeur repasse en attente



Basic Communication :

Pour avancer le robot nécessite un envoi d'un signal qui est une suite de caractères codés en char.

Le premier char est composé du nombre : « 255 ». Le deuxième char est la taille, dans le cas de Wifibot, le plus souvent c'est 0x07 pour la décimal 7.

Pour le troisième et quatrièmes, c'est la vitesse de rotation de gauche que l'on fait varier de 0 -> 240 « tics » qui sont une unité définie dans la doc.

Pour le cinquième et sixième, c'est la vitesse de rotation de droite que l'on fait varier de 0 -> 240 « tics ».

Le septième est l'orientation de rotation des roues et le ON/OFF des roues.

Le char est décomposé de la manière suivante (1 bytes char -> 8 bits)

(128) Bit 7 Contrôle de vitesse en boucle fermée côté gauche : 1 -> ON / 0 -> OFF

(64) Bit 6 Signal Indicateur de vitesse avant/arrière côté gauche : 1 -> avant / 0 -> arrière (32) Bit 5

Contrôle de vitesse en boucle fermée côté droite : 1 -> ON / 0 -> OFF

(16) Bit 6 Signal Indicateur de vitesse avant/arrière côté droite : 1 -> avant / 0 -> arrière

(8) Bit 3 Relais 4 On/Off (D'SUB 15 POWER Pin 13 et 14)

(4) Bit 2 Relais 3 On/Off (D'SUB 15 POWER Pin 11 et 12)

(2) Bit 1 Relais 2 On/Off (D'SUB 15 POWER Pin 4 et 5)

(1) Bit 0 Relais 1 pour capteur. On/Off : 0 est OFF 1 est ON (DSUB15 POWER Pin 3)

Les huitième et neuvième Char sont le CRC 16 bits.

Par exemple, pour contrôler le côté gauche et le côté droit (Gauche & Droite 2 moteurs, 2 encodeurs) : La vitesse est comprise entre 0-240. Si nous voulons que le côté gauche et le côté droit se déplacent à la vitesse 120 vers l'avant sans contrôle moteur, nous envoyons :

Char 1 : 255

Char 2 : 0x07

Char 3 : 120

Char 4 : 00

Char 5 : 120

Char 6 : 0

Char 7 : 80 ($0 + 64 + 0 + 16$)

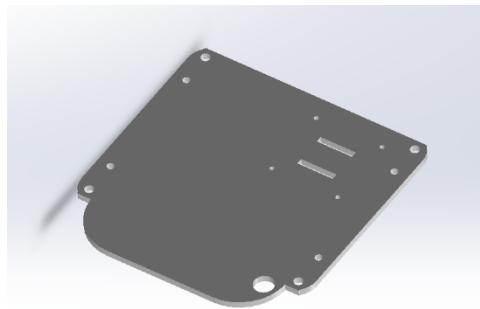
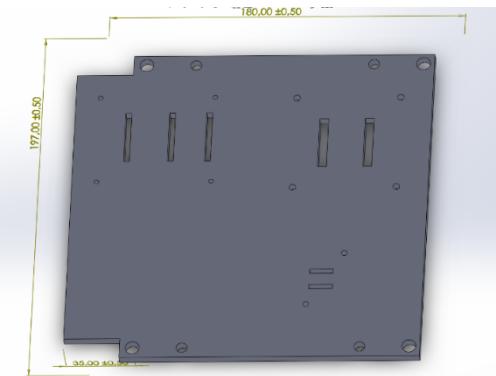
Char 8 – Char 9 : CRC16(data)

Commande du WifiBot avec ROS

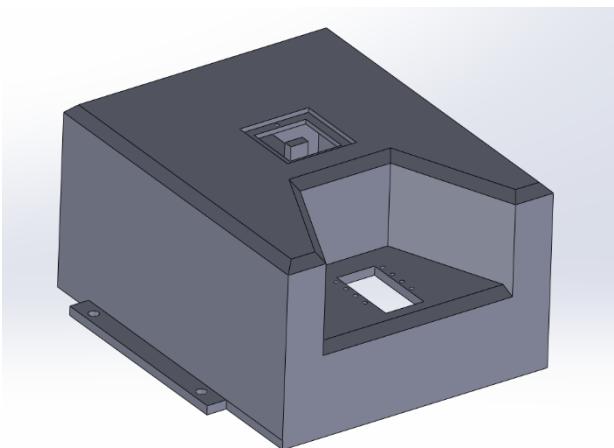
Modélisation 3D

Afin d'optimiser la réalisation du robot, de nombreuses parties ont été conçues en 3D avant la phase de conception physique. Trois versions ont été développées pour chaque composant : la première en tant que prototype, la deuxième en tant que version fonctionnelle permettant des tests et du prototypage, et enfin la version finale, à la fois esthétique et fonctionnelle.

La première composante est le socle, qui se fixera sur le WifiBot, abritant les emplacements pour les cartes électroniques et renforçant certaines parties du bras robotique. Ci-dessous, vous trouverez la version 2 du support de carte ainsi que la nouvelle version de ce dernier :

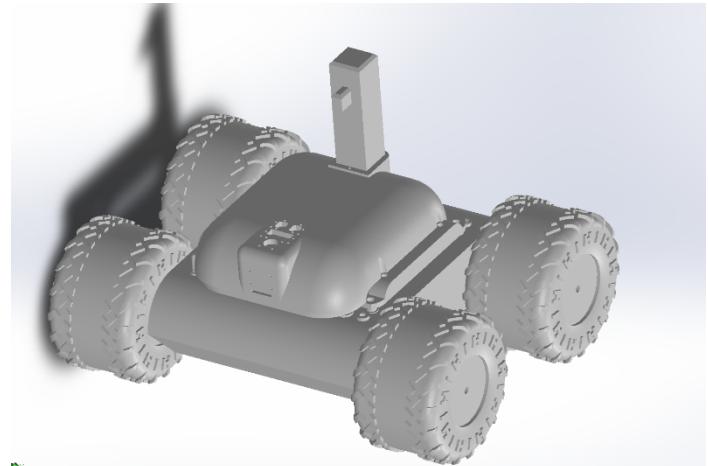


La seconde pièce est le capot du robot. La V2 se caractérise par sa simplicité et sa forme carrée et haute, conçue pour accueillir tous les composants à l'intérieur afin de les protéger de l'environnement extérieur. En revanche, la V3 est plus basse, plus arrondie et ergonomique, offrant une esthétique améliorée. Voici les deux versions des capots :



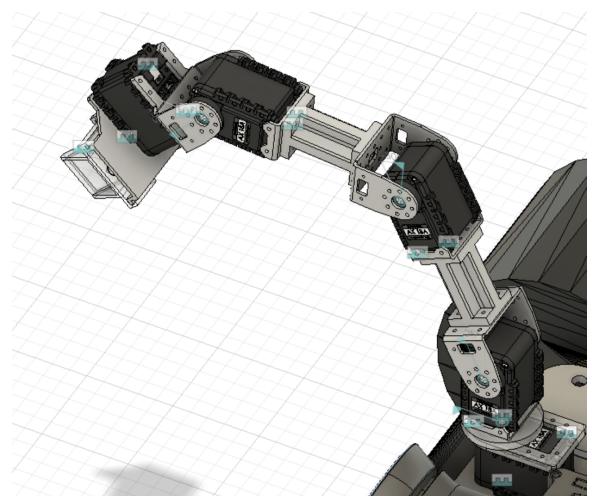
Enfin, la dernière partie est la tour destinée à accueillir la caméra et le capteur Lidar, permettant la cartographie de l'environnement où se trouve le robot.

Une fois assemblée, la structure du robot présente l'aspect suivant :

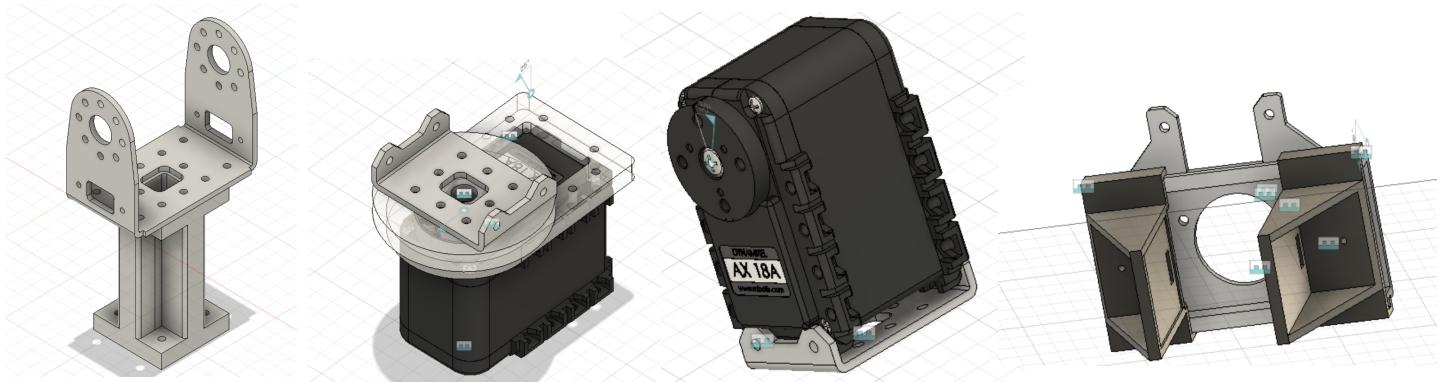


Pour mener à bien le projet et accomplir les tâches qui lui sont assignées, il était impératif de concevoir un bras robotique adaptable au WifiBot.

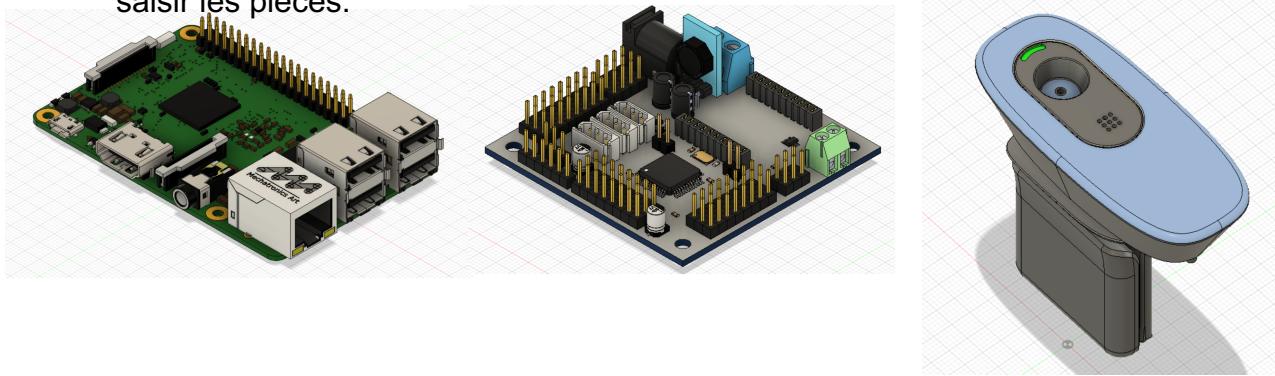
La première étape a consisté à déterminer le type de manipulateur pouvant être utilisé pour le projet. Celui-ci devait simplement être capable de soulever une petite charge, compte tenu de son stade de prototype.



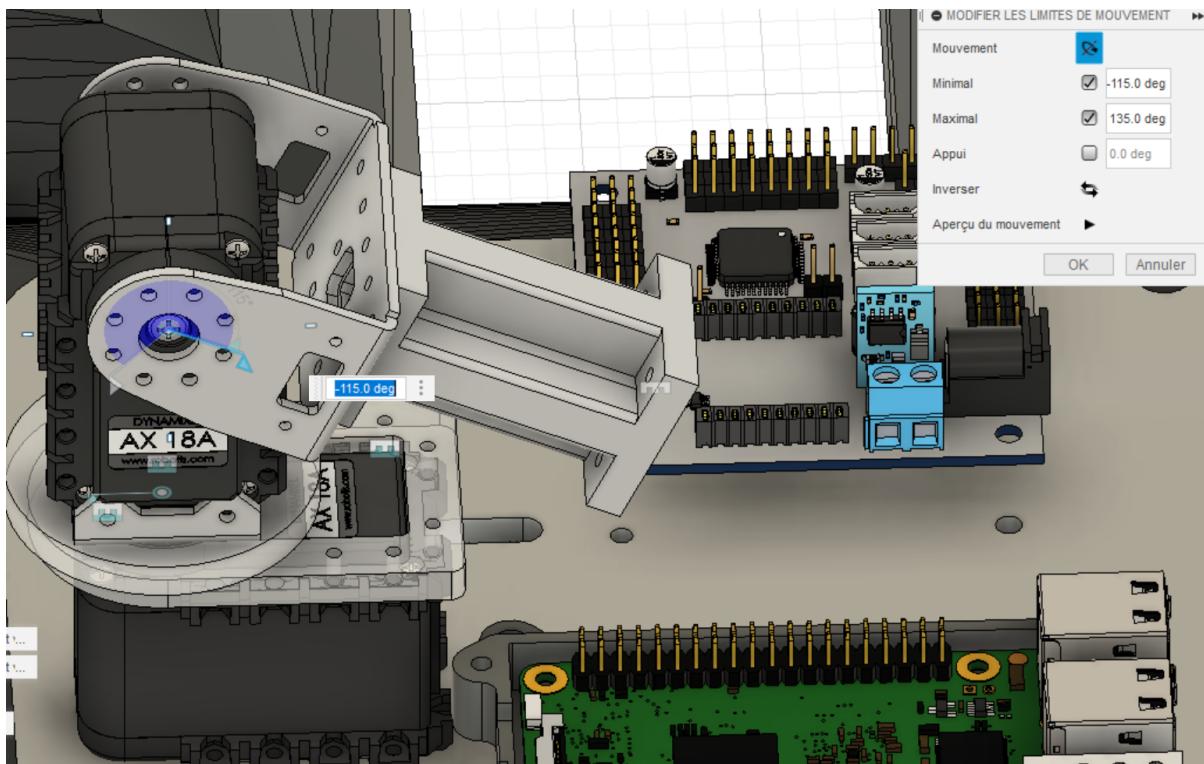
Il était donc essentiel de s'inspirer de ce qui était réalisé dans le monde industriel, en particulier en ce qui concerne les bras manipulateurs utilisés sur les "rovers" professionnels. La conclusion a été qu'un manipulateur à 5 axes serait suffisant pour le projet.



Ainsi, le manipulateur à 5 axes a été conçu sur Fusion 360, incluant la modélisation des segments, des servomoteurs, de leurs supports, ainsi que du préhenseur permettant de saisir les pièces.



Pour obtenir une vision plus précise du robot dans le monde réel, avec l'emplacement des composants et les contraintes de mouvement du manipulateur, tous les éléments ont été inclus dans notre modélisation. Cette approche nous a permis de définir les limites angulaires de notre manipulateur.



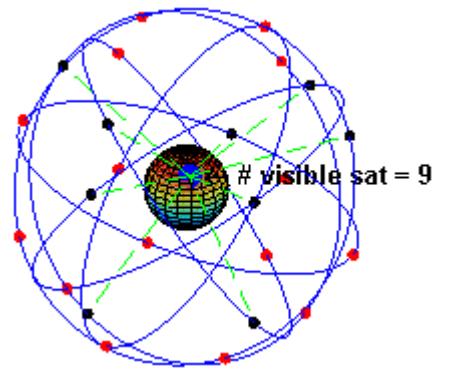
GPS

Dans le cadre du projet qui a pour objectif de se déplacer en milieu hostile et être commandé à distance il y a de forte chance pour que le droit ne soit plus visible. La solution est un module GPS

Principe de fonctionnement :

Le Global Positioning System (en français : « Système mondial de positionnement » ou « Géo-positionnement par satellite »), est un système de positionnement par satellites.

Le GPS fonctionne grâce au calcul de la distance qui sépare un récepteur GPS de plusieurs satellites. Les informations nécessaires au calcul de la position des satellites étant transmises régulièrement au récepteur, celui-ci peut, grâce à la connaissance de la distance qui le sépare des satellites, connaître ses coordonnées. Les récepteurs GPS utilisent ces informations et la trilateration pour calculer la position exacte d'un utilisateur.



Les points représentent des satellites, en noir les satellites qui se charge de la localisation de ce GPS en rouges d'autre inactifs pour ce GPS.

Les recherches nous ont mené à ce GPS le Module GPS GY-NEO6MV2 de géolocalisation. Il est compatible avec la carte Raspberry Pi et coûte 10 euros :

Quelques spécifications techniques :

- Taille Module : 25mm x 25mm
- Taille Antenne : 36mm x 26mm
- Alimentation : 3.3V DC - 5V DC
- Liaison série : 9600bps
- Sensibilité antenne : -161 dBm

Branchements :

VCC à la broche 1, qui est 3.3v

RX à la broche 10, qui est RX (GPIO15)

TX à la broche 8, qui est TX (GPIO14)

Gnd à la broche 6, qui est Gnd.

ne fois les connections effectuer voici comment lancé le GPS et l'utiliser. Par défaut, le Raspberry Pi utilise l'UART comme console série. Il faut désactiver cette fonctionnalité afin de pouvoir utiliser l'UART pour notre propre application. Ouvrez une session de terminal sur le Raspberry Pi. La première chose à faire est de sauvegarder le fichier cmdline.txt avant de le modifier.

`sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt` et appuyez sur Entrée.

Ensuite, éditer cmdlint.txt et supprimer l'interface série.

Tapez `sudo nano /boot/cmdline.txt` et appuyez sur Entrée.

Supprimez `console=ttyAMA0,115200` et sauvegardez le fichier en appuyant sur Ctrl X, Y et Enter.

Tapez maintenant `sudo nano /etc/inittab` et appuyez sur Entrée.

Trouvez `ttyAMA0` en appuyant sur Ctrl W et en tapant `ttyAMA0` sur la ligne de recherche.

Lorsqu'il trouve cette ligne, appuyez sur home, insérez un symbole # pour commenter cette ligne, et appuyez sur Ctrl X, Y, Enter pour sauvegarder. Tapez `sudo reboot` et appuyez sur Entrée pour redémarrer le Pi.

Raspberry Pi B+ J8 Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	I2C GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	BITCLOCK_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	SPI GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	DATA IN	38
39	Ground	DATA OUT	40

Time:	2015-07-26T16:14:01.000Z	PRN:	Elev:	Azim:	SNR:	Used:
Latitude:	33.458175 N	1	41	146	26	Y
Longitude:	79.566635 W	4	47	108	28	Y
Altitude:	-41.0 ft	7	86	228	08	Y
Speed:	0.5 mph	8	13	047	32	Y
Heading:	0.0 deg (true)	9	17	205	21	Y
Climb:	0.0 ft/min	11	63	138	21	Y
Status:	3D FIX (537 secs)	13	07	321	00	N
Longitude Err:	+/- 65 ft	16	00	089	00	N
Latitude Err:	+/- 53 ft	17	08	231	18	N
Altitude Err:	+/- 108 ft	19	44	040	21	N
Course Err:	n/a	27	13	046	29	N
Speed Err:	+/- 89 mph	28	29	296	11	N
Time offset:	0.575	30	55	314	00	N
Grid Square:	FM03fk					

Tester le GPS en utilisant quelques programmes standard. Ouvrez une session de terminal et tapez sudo apt-get install gpsd gpsd-clients et appuyez sur Entrée. Post l'installation, il faut démarré sur le port série :

Tapez stty -F /dev/ttyAMA0 9600 et appuyez sur Entrée.

Démarrez maintenant GPSD :

Tapez sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock et appuyez sur Entrée.

Affichez maintenant en tapant cgps -s et appuyez sur Entrée.

Cartographique

Le RPLIDAR est un capteur LIDAR peu coûteux adapté aux applications de SLAM robotique en intérieur. Il offre un champ de balayage à 360 degrés, une fréquence de rotation de 5,5 Hz/10 Hz avec une distance garantie de 25 mètres pour le modèle A3.

Grâce au moteur de traitement d'image à haute vitesse conçu par RoboPeak, les coûts globaux sont considérablement réduits, faisant du RPLIDAR le capteur idéal dans des domaines sensibles aux coûts tels que les robots grand public et les amateurs de matériel.

Le RPLIDAR A3 effectue une mesure de distance à grande vitesse avec plus de 16 000 échantillons par seconde. Pour un balayage nécessitant 360 échantillons par rotation, une fréquence de balayage de 10 Hz peut être atteinte. Les utilisateurs peuvent personnaliser la fréquence de balayage de 2 Hz à 10 Hz librement en contrôlant la vitesse du moteur de balayage. Le RPLIDAR s'auto-adapte à la vitesse de balayage actuelle

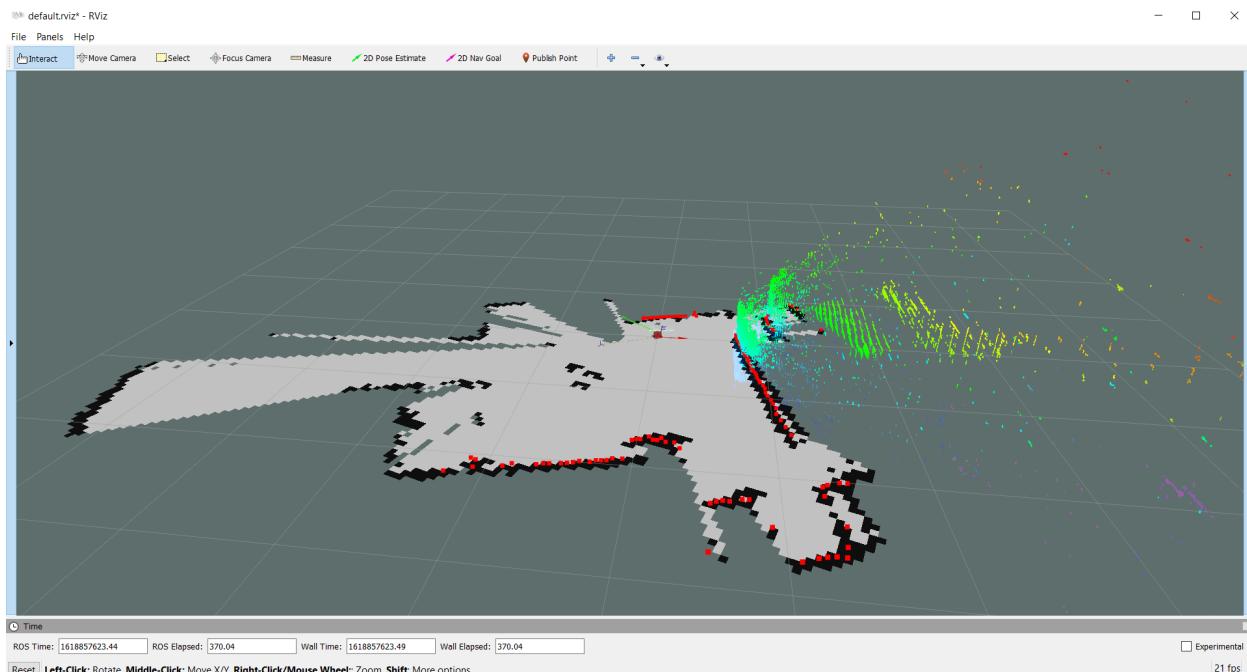
Le RPLIDAR A3 effectue une mesure de distance à grande vitesse avec plus de 16 000 échantillons par seconde. Le RPLIDAR s'auto-adapte à la vitesse de balayage actuelle.



La création d'une carte à l'aide d'Hector SLAM sous ROS 2 avec un capteur LIDAR A3 offre une solution robuste et efficace pour la cartographie en temps réel dans des environnements robotiques. ROS 2, successeur de ROS (Robot Operating System), introduit des améliorations significatives en termes de modularité, de performance et de prise en charge des nouvelles fonctionnalités matérielles. Hector SLAM, quant à lui, est un algorithme SLAM (Simultaneous Localization and Mapping) qui permet au robot de se localiser et de cartographier son environnement simultanément.

Démarré le programme : Exécutez le système en utilisant la commande de lancement appropriée. Assurez-vous que le LIDAR A3 est alimenté, et le robot est libre de se déplacer dans l'environnement que vous souhaitez cartographier.

ros2 launch hector_slam_launch mapping_launch.py



Pendant que le robot se déplace, le LIDAR A3 collecte en continu des données de distance. Hector SLAM utilise ces données pour estimer la pose du robot et créer une carte en temps réel. La carte résultante est mise à jour à mesure que le robot explore de nouveaux secteurs de l'environnement.

Ont utiliser des outils de visualisation ROS 2, Rviz, pour visualiser la carte en temps réel.

Le programme est un mixte du programme python permettant de faire tourner le Capteur lidar ainsi que du programme GitHub Hector Slam.

Partie Retour vidéo

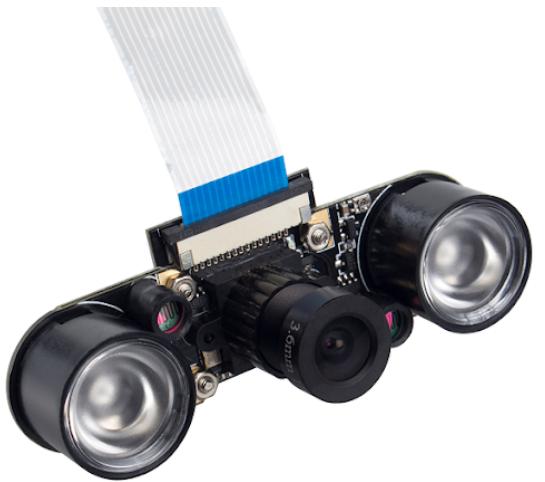
L'utilisation d'une caméra Raspberry Pi avec vision nocturne offre des fonctionnalités étendues pour la capture d'images dans des conditions de faible luminosité. En outre, l'intégration d'une deuxième caméra permettant de générer deux flux vidéo en temps réel pour une diffusion sur une page web offre une solution polyvalente pour la surveillance, la sécurité ou d'autres applications nécessitant une surveillance à distance.

La caméra Raspberry Pi équipée de vision nocturne utilise généralement des capteurs infrarouges pour capturer des images même dans l'obscurité. Ces capteurs activent automatiquement des LED infrarouges pour éclairer la scène, créant ainsi des images claires et détaillées même dans des conditions de faible éclairage. Cette capacité s'avère précieuse dans divers contextes, tels que la surveillance nocturne ou la capture d'images dans des endroits peu éclairés.

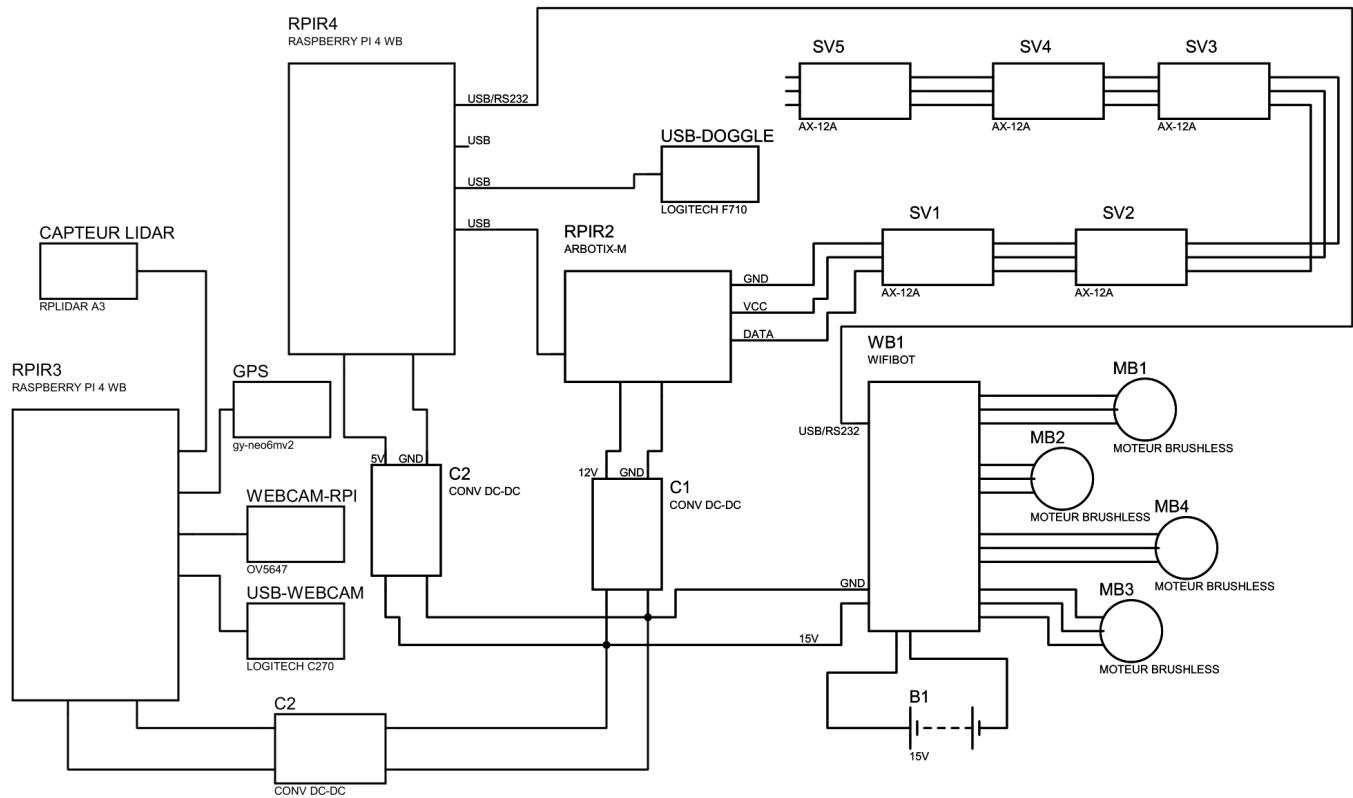
Pour créer un système avec deux flux vidéo en direct, vous pouvez utiliser une deuxième caméra Raspberry Pi standard. Ces caméras peuvent être configurées pour générer des flux vidéo distincts et être connectées à la Raspberry Pi via les ports dédiés. Il a été utilisé le logiciel tel que Motion qui permet gérer la diffusion des flux vidéo vers une page web.

Une première caméra Raspberry Pi à un port USB et la deuxième utilise le port dédié pour une caméra CSI (Camera Serial Interface) sur la Raspberry Pi.

Une fois connecté à la Raspberry Pi, les deux flux des caméras sont récupérés par Motion et transférés sur une page web. Nous avons ajouté le module Rviz ainsi que



Montage électronique



Partie programmation vision

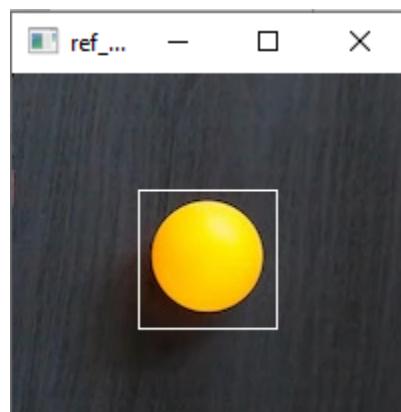
La possible utilisation d'une carte graphique comme carte embarquée pour une meilleure performance pourrait être envisageable, mais nous avons finalement choisi de garder la Raspberry car plus compatible avec les autres modules du robot, le programme doit donc être le moins gourmand possible pour toujours assurer la bonne vision et un frame rate suffisant.

Les différentes étapes de la programmation ont été les suivantes :

- Recherches YOLOv5 (trop gourmand en ressources pour une Raps) / utilisation d'open CV ou cv2 possible / créer un programme léger pour tourner efficacement car il devra combiner la vue en temps réelle du robot ainsi que la détection et l'interprétation de l'image observée



- Détection sur une photo d'objet



- Détection de bords de l'objet

```
frame = imutils.resize(frame, width=600)
blurred = cv2.GaussianBlur(frame, (11, 11), 0)
hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

mask = cv2.inRange(hsv, ColorLower, ColorUpper)
mask = cv2.erode(mask, None, iterations=2)
mask = cv2.dilate(mask, None, iterations=2)
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

cnts = imutils.grab_contours(cnts)
```

- Création d'un cercle afin de s'en servir ensuite pour trouver la distance caméra/objet

```
if len(cnts) > 0:

    c = max(cnts, key=cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)
    M = cv2.moments(c)
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

    if radius > 15 :

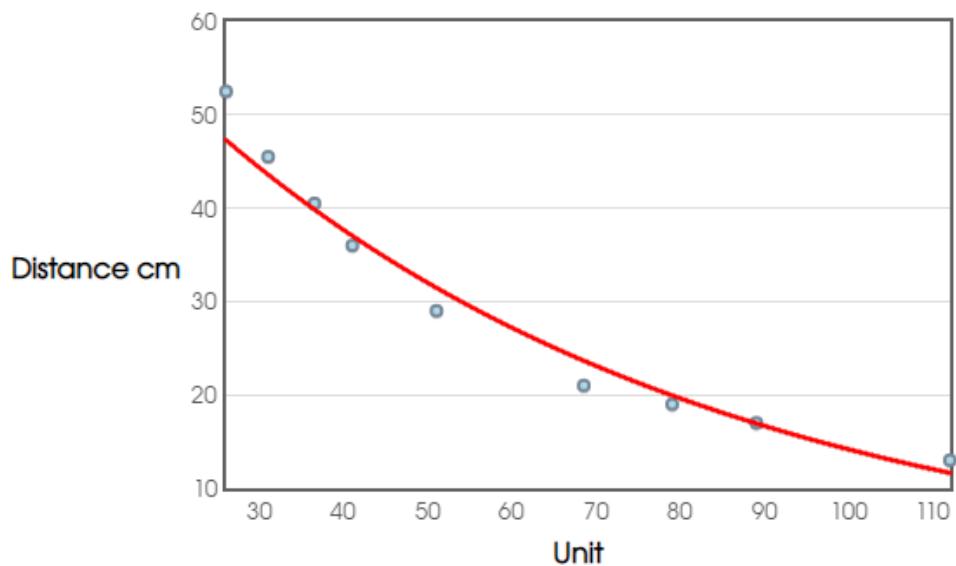
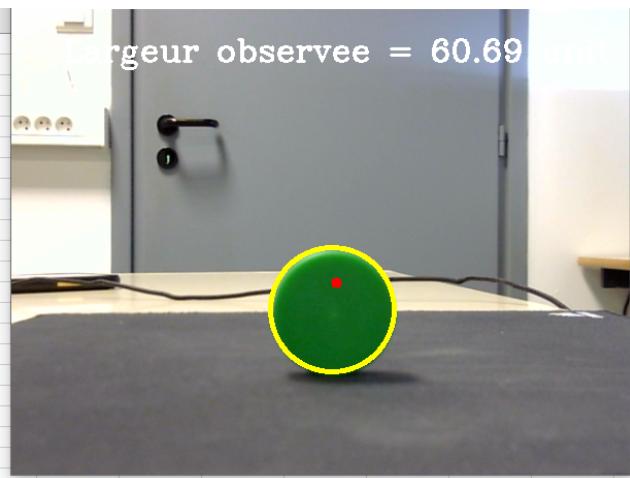
        cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 3)
        cv2.circle(frame, center, 5, (0, 0, 255), -1)

        if radius != 0 :
            cv2.putText(frame, f"Largeur observee = {round(radius,2)} unit", (50, 50), fonts, 1, (WHITE), 2)

    pts.appendleft(center)
```

- Définition d'une formule pour trouver la bonne distance entre la caméra et l'objet grâce à des tests

A	B	C	D	E	F	G	H	I
distance réelle		21	36	40,5	45,5	29	52,5	
Units		68,5	41	36,5	31	51	26	



Regression Equation: $\text{DISTANCE CM} = 72.24 \cdot 0.98^{\text{UNIT}}$

```
f"Distance = {round(72.24*0.98**int(radius),2)} cm"
```

- Mise en place du module sur python pour la détection vidéo (en temps réel)

```
if not args.get("video", False):
    vs = VideoStream(src=0).start() #set up usb port

else:
    vs = cv2.VideoCapture(args["video"])

time.sleep(2.0)
```

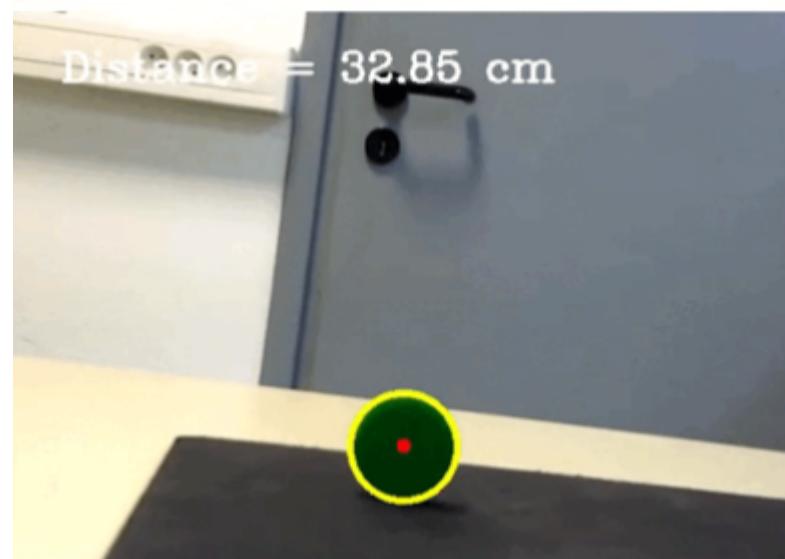
- Comparaison sur cv2 pour un fichier xml (grosse base de données)

```
ball_detector = cv2.CascadeClassifier("C:/Users/33778/Documents/Cours_Esirem/4A/Projet_4A/CodesPythonXml/rawballdetection.xml")
```

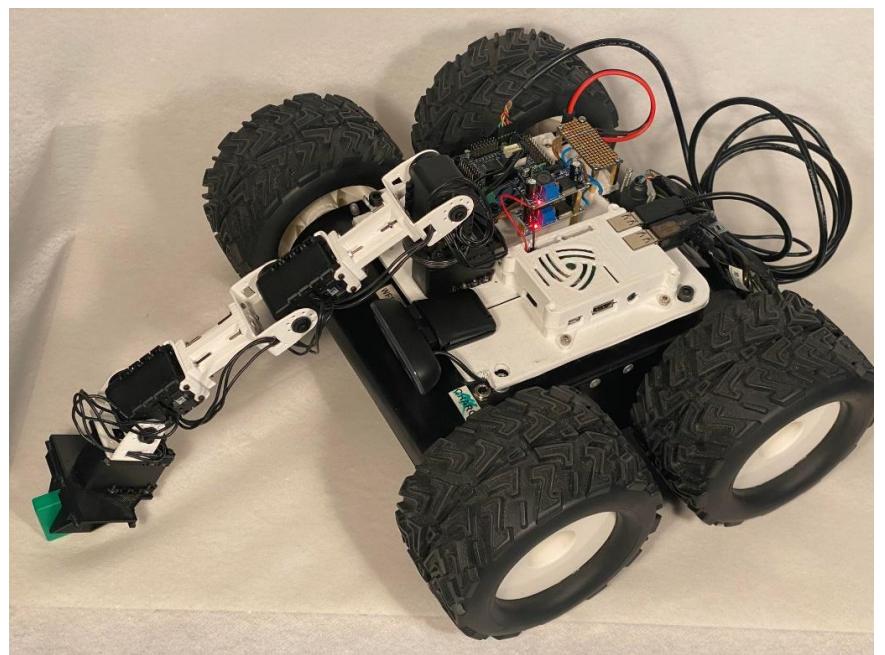
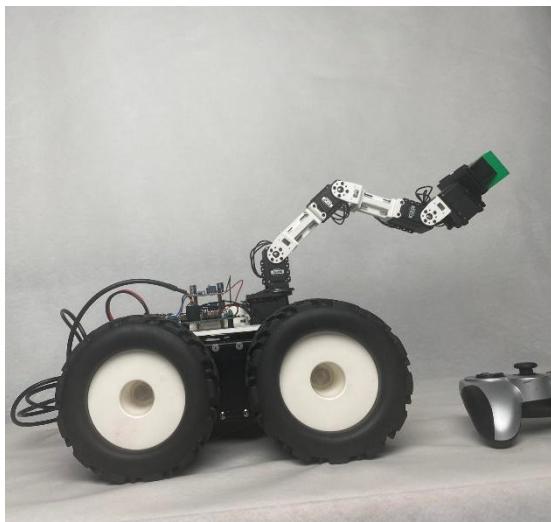
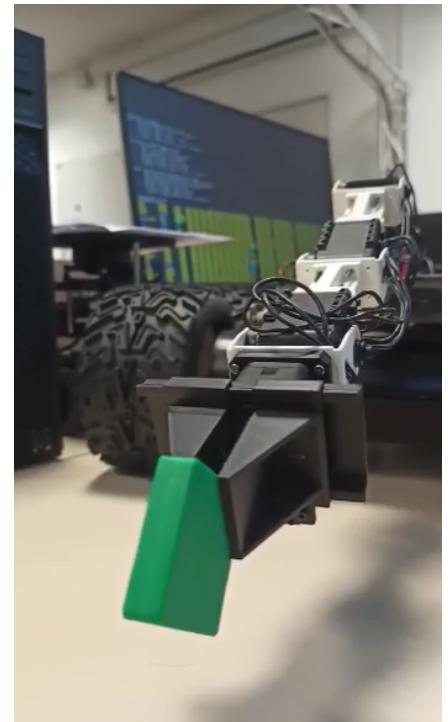
- Adaptation du programme pour sa légèreté avec simplement détection finale d'une balle de couleur.

```
ColorLower = (29, 86, 6)      #red#(160,100,20)
ColorUpper = (64, 255, 255)    #red#(179,255,255)
```

Version finale du programme transférée et adaptée à la Raspberry, où le frame rate est largement suffisant pour notre utilisation et où la valeur distance-objet peut être communiquée directement au bras robot.



Mise en commun et assemblage



Quelles optimisations possibles ?

Les optimisations possibles du robot pourront être :

1. La mise en place d'un système de commande sans fil performant afin de commander le robot sur une très grande distance
2. L'ajout d'un deuxième bras robotisé afin de permettre la manipulation d'objets de plus grande taille.
3. La création d'un logiciel pour le déplacement autonome pour le repenser en robot mobile.
4. Zone de stockage sur le robot pour récupérer plus de déchets.

Difficultés :

La première et principale difficulté a été la connexion avec le WifiBot. En effet, le code fourni dans la documentation technique était adapté à la carte mère vendue avec le robot. Malheureusement dans notre cas, il a fallu utiliser un Raspberry Pi et nous orienter plus vers un système tournant sous Ros.

Le seconde problème est que le Raspberry Pi a du mal à faire tourner l'intégralité du système correctement.

Avis personnel

Joshua Theron : Ce projet a été un réel plaisir à suivre, c'était une idée que je voulais réaliser de longue date. Il m'a personnellement beaucoup apporté car j'ai pu découvrir le domaine des drones terrestres et leur fonctionnement. La partie où j'ai le plus appris a été la programmation du Lidar qui ma apporter de nouvelle compétence

Gregory Davillars : N'ayant jamais pu travailler sur Raspberry ou ROS, ce projet fut très enrichissant, me permettant d'acquérir des connaissances dans des domaines que je ne connaissais pas. Il m'a aussi permis de renforcer mes bases en linux et en Bash, la connexion entre le WifiBot et mon PC se faisant en ssh et le script de contrôle en Bash.

Bastien Berland : Projet enrichissant, une vraie envie de le mettre en œuvre et de le concrétiser, réflexion approfondie lors de l'établissement du cahier des charges et des diagrammes. Sujet remplis de difficultés techniques qui apportent sont lots de solutions intéressantes.

Remerciements

Nous tenons tout d'abord à remercier Joaquim Rodriguez qui nous a été d'une grande aide lors de la mise en route du WifiBot, mais aussi Raphaël Duverne pour le prêt matérie. Nous souhaitons également adresser nos remerciements les plus sincères aux équipes pédagogiques et administratives de l'Esirem ainsi que de l'IUT qui ont facilité nos démarches et ont assuré un suivi quand nous le désirions tout au long de notre projet.

Conclusion :

Après de longues semaines de recherches et de mise en place, nous avons enfin pu créer une maquette fonctionnelle. Le Robot d'exploration a été un projet des plus intéressants, qui nous a donné du fil à retordre. Il nous a permis de découvrir beaucoup de choses au niveau électrique, mécanique et informatique et a pu renforcer notre cohésion et travail d'équipe.

Sources

<https://www.wifibot.com/>

<https://github.com/search?l=C%2B%2B&q=wifibot&type=Repositories>

<https://www.freva.com/assign-fixed-usb-port-names-to-your-raspberry-pi/>

<https://arduino103.blogspot.com/2017/10/raspberry-pi-detection-du-materiel-et.html>

<https://blog.mbedded.ninja/programming/operating-systems/linux/linux-serial-ports-using-c-cpp/#overview>

https://github.com/radosz99/tennis-ball-detector/blob/main/classifiers/cascade_12stages_24dim_0_25far.xml

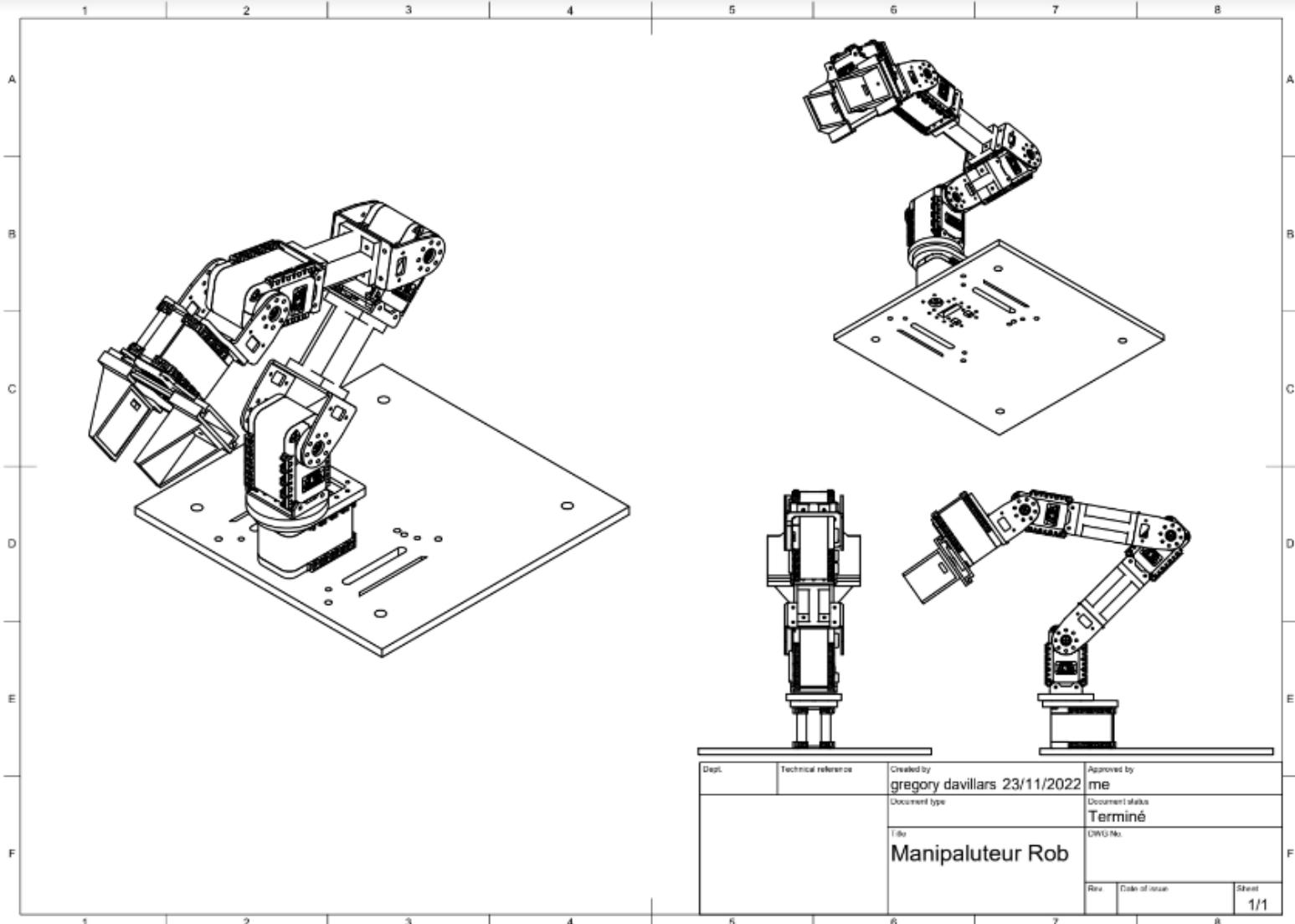
<https://pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/>

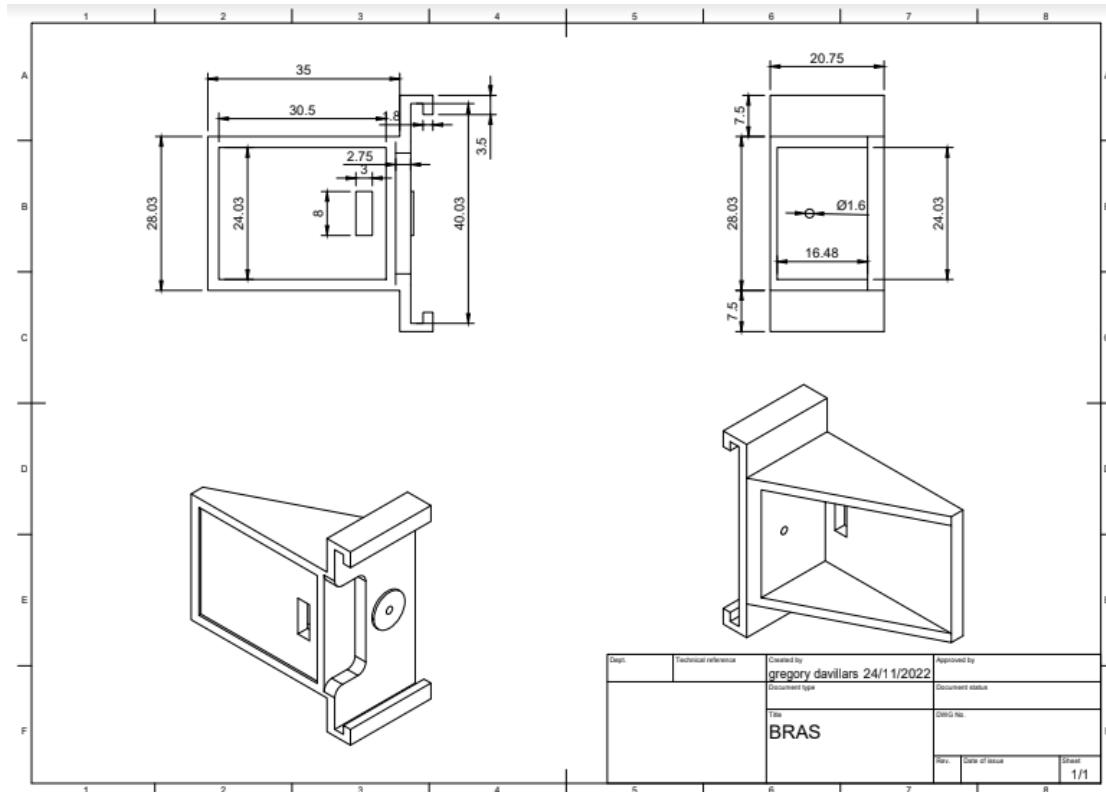
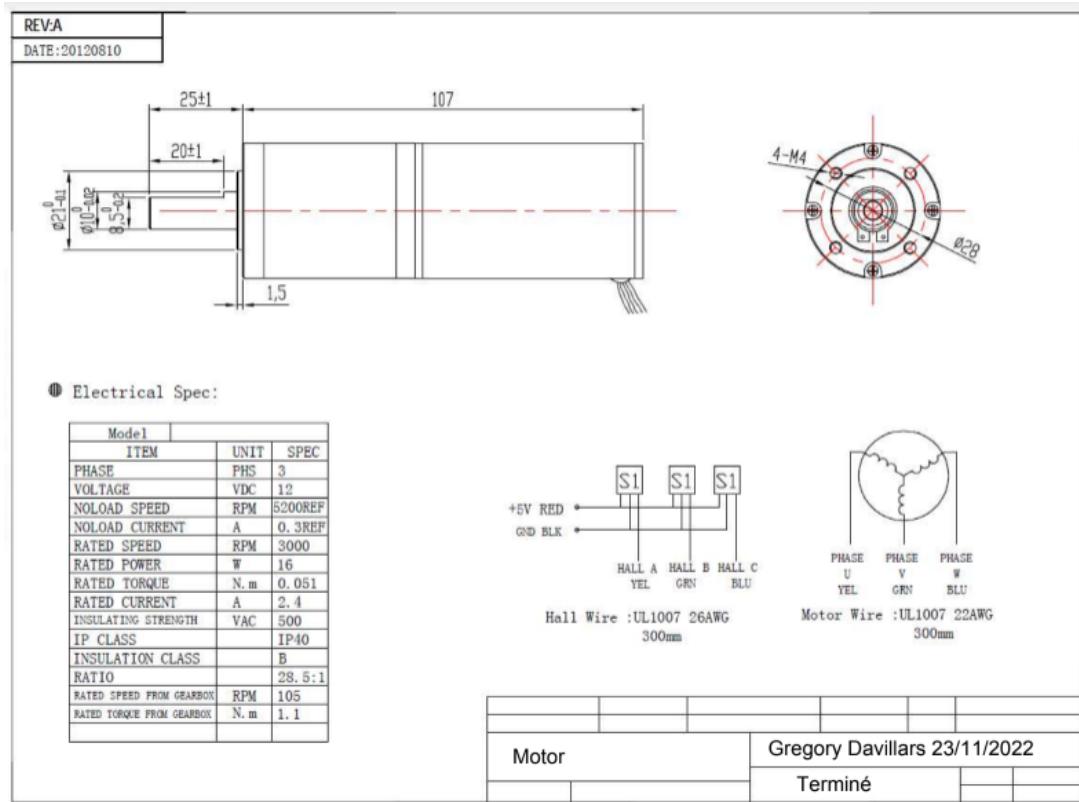
https://github.com/kipr/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml

<https://bitbucket.org/Joko1991/code-wifibot/src/master/>

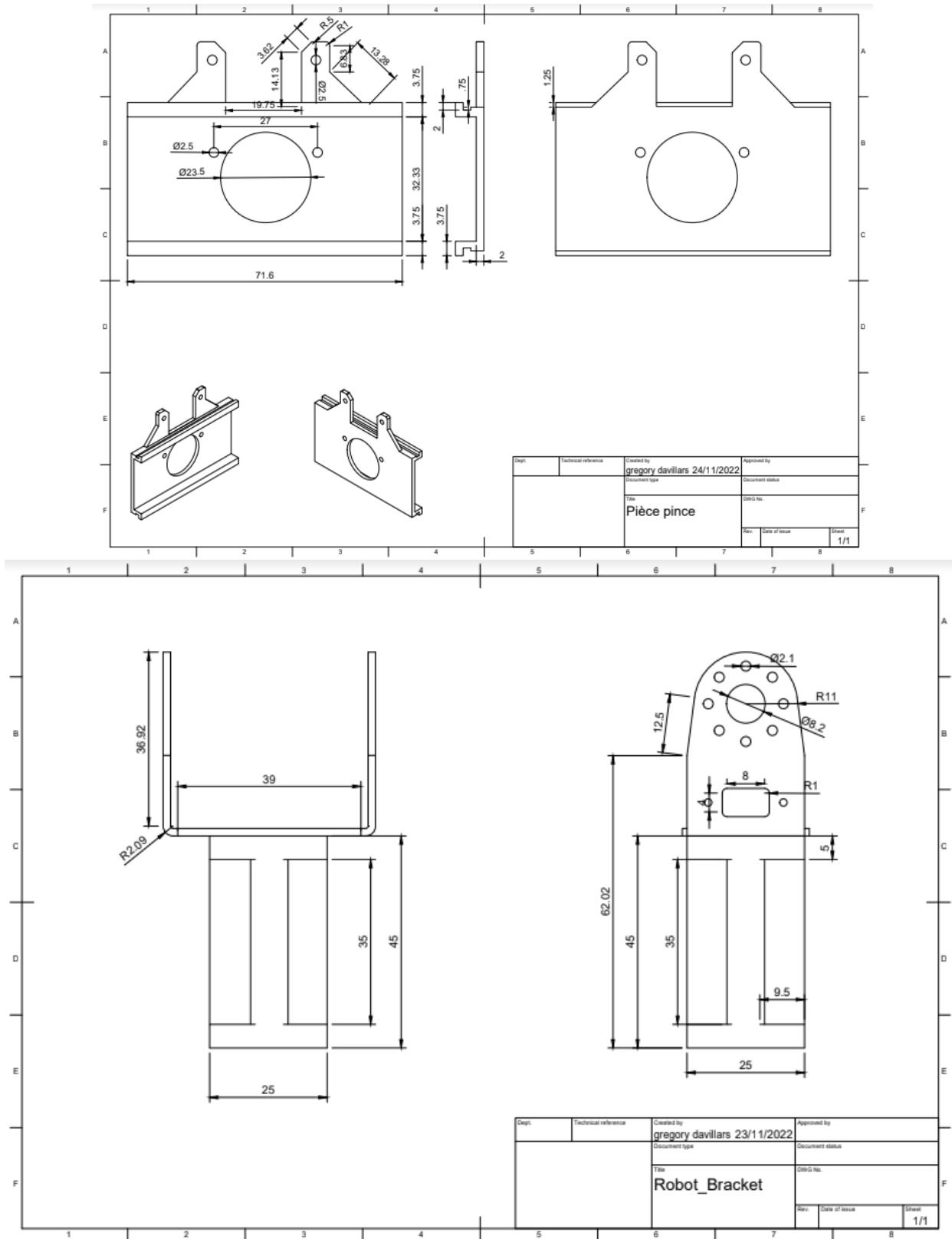
Annexes

Plan de définition

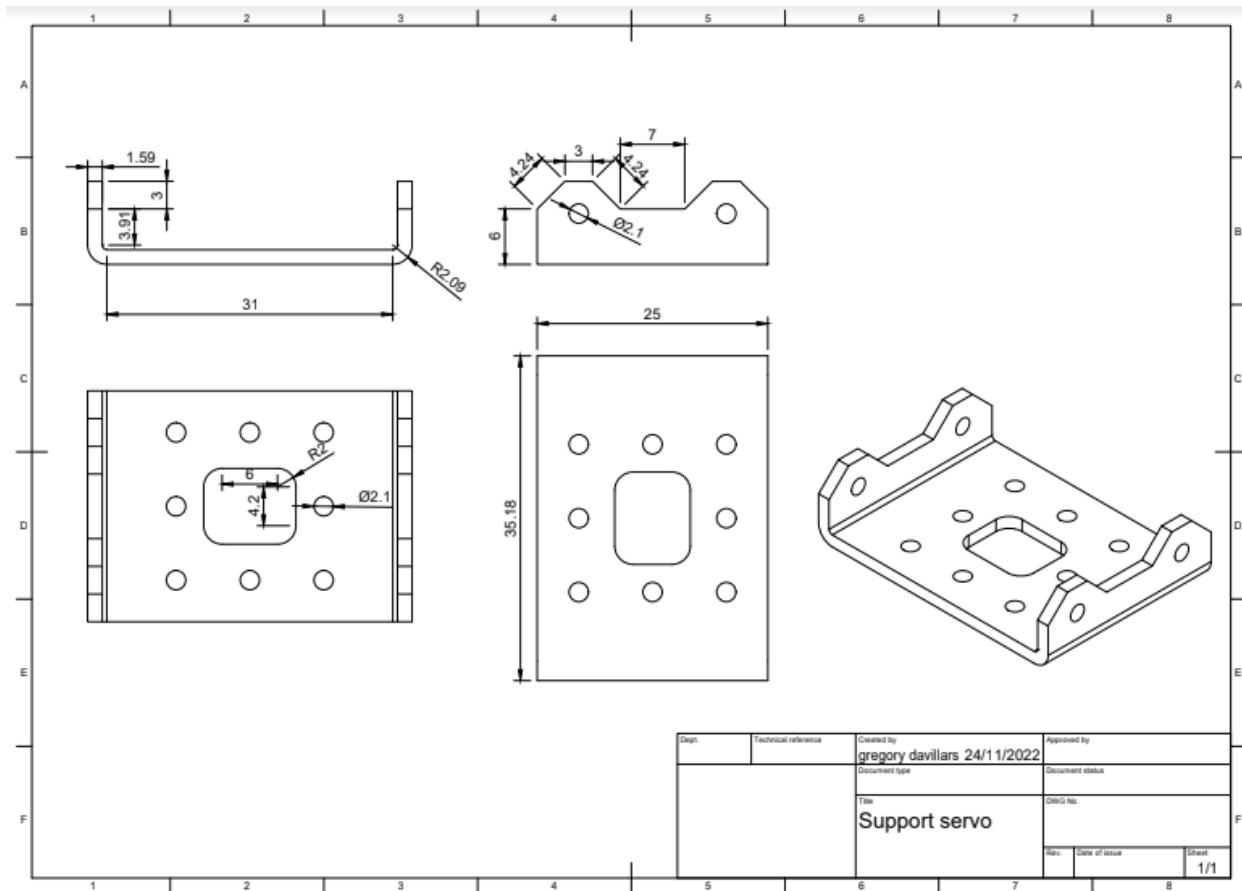


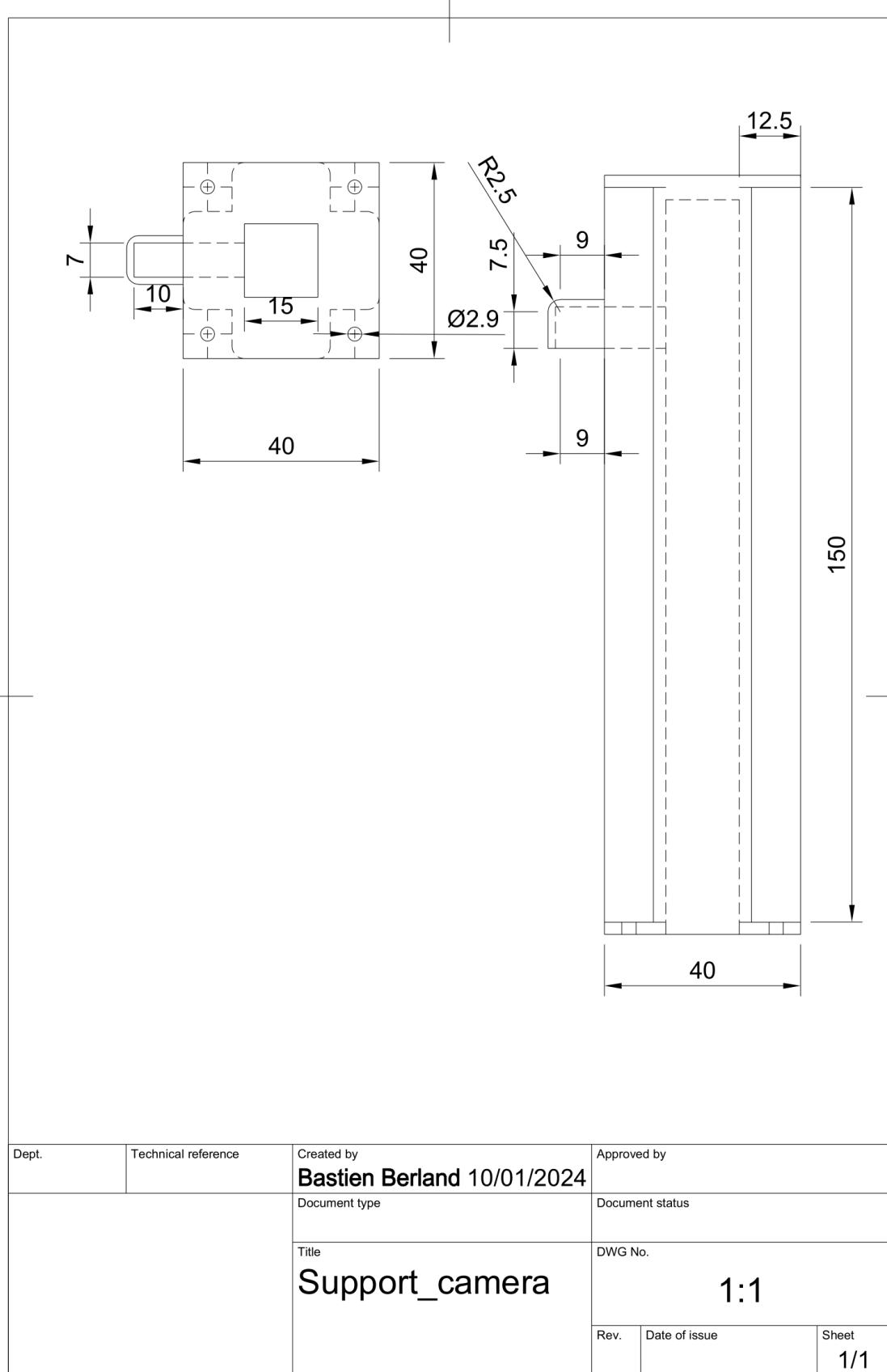


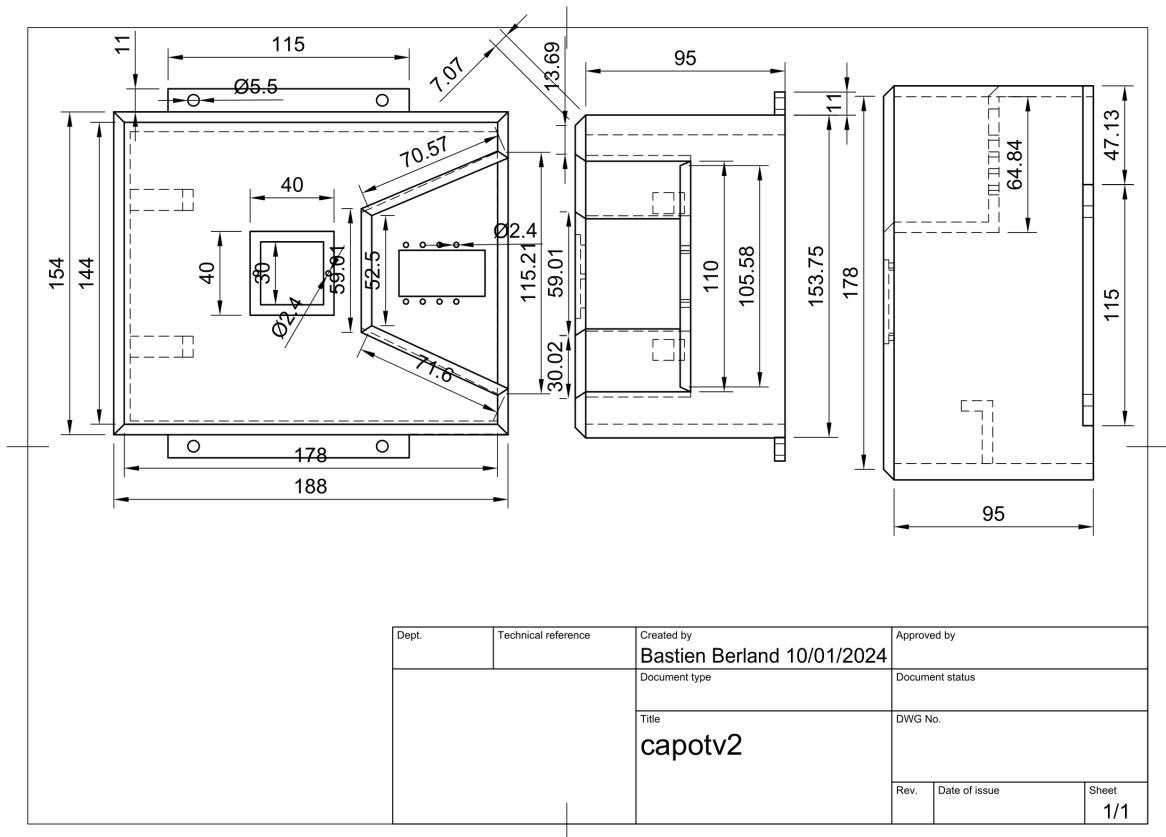
Projet tutoré 2023-2024

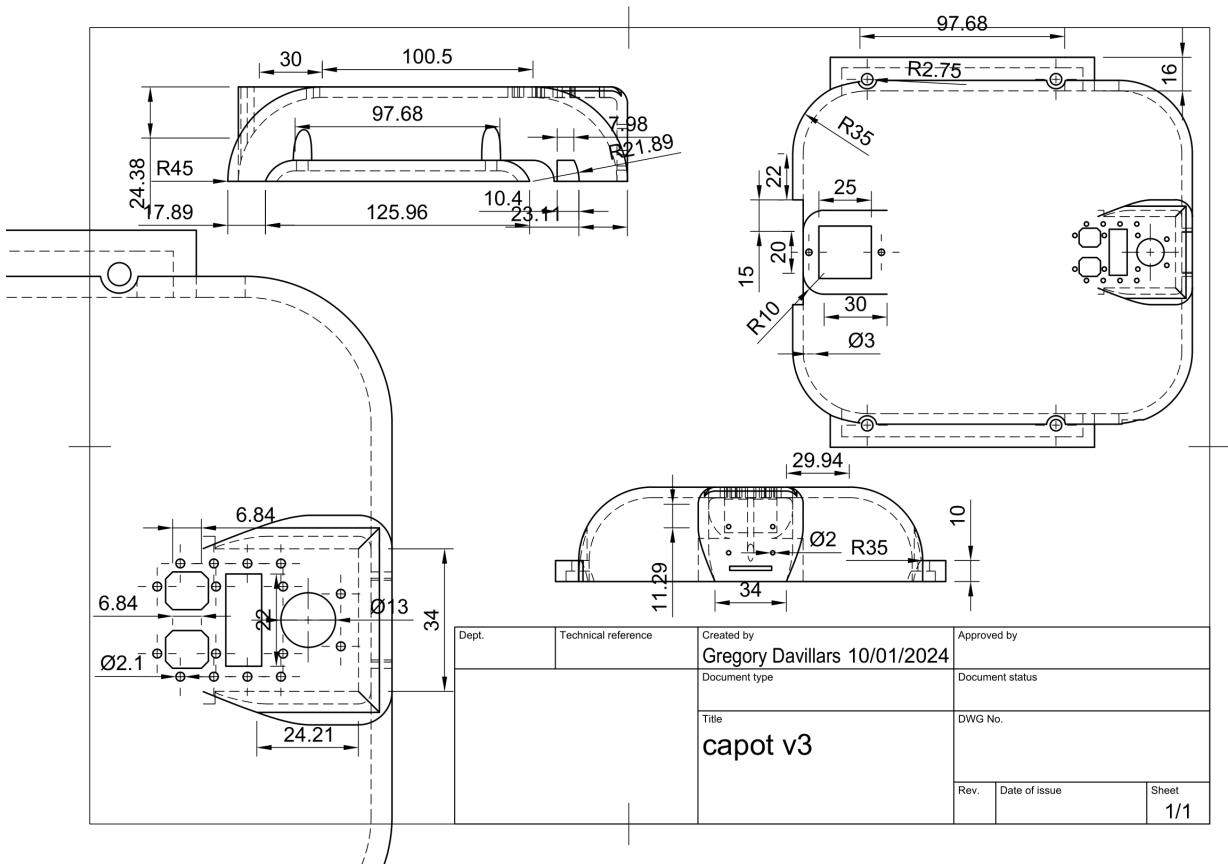


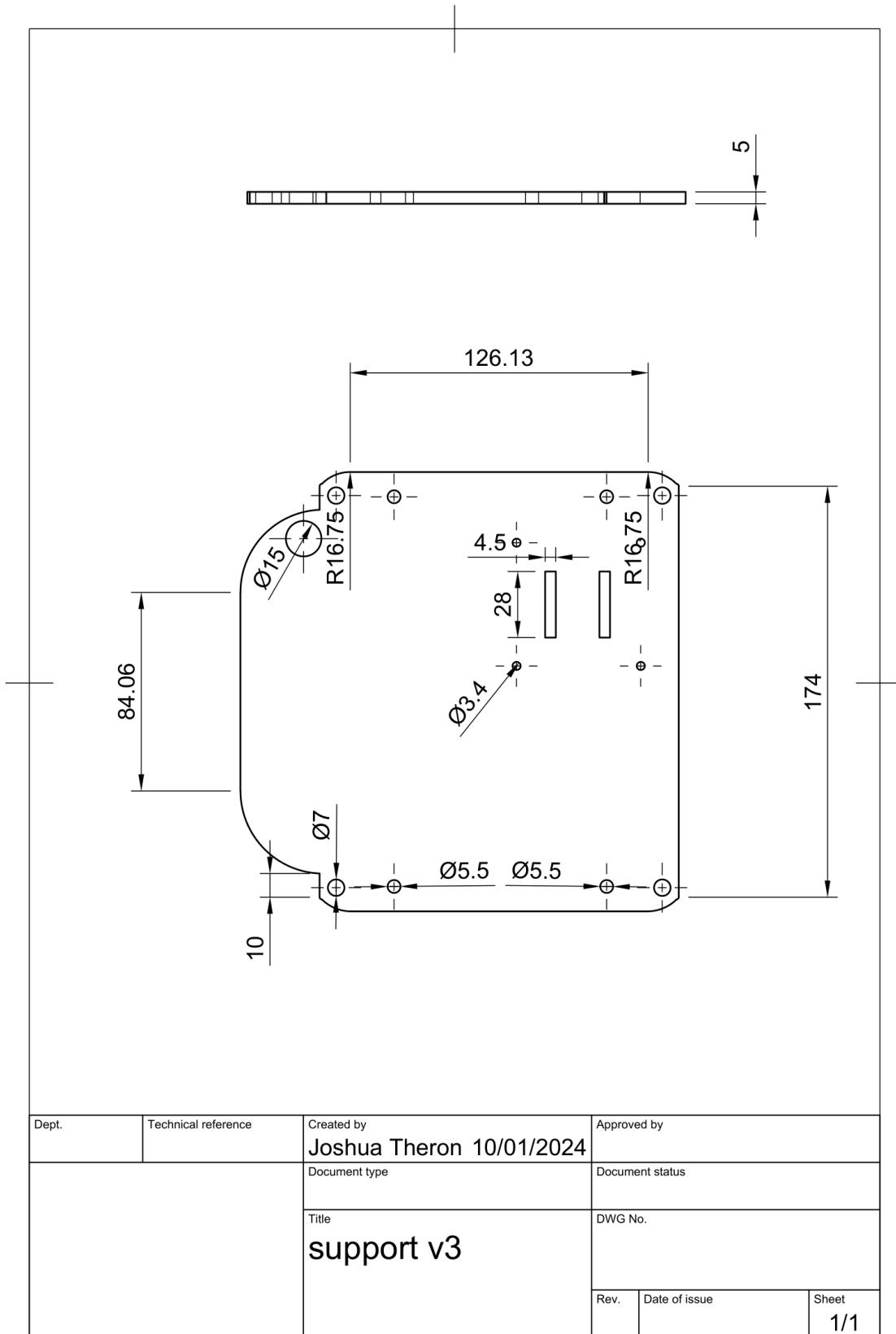
Projet tutoré 2023-2024











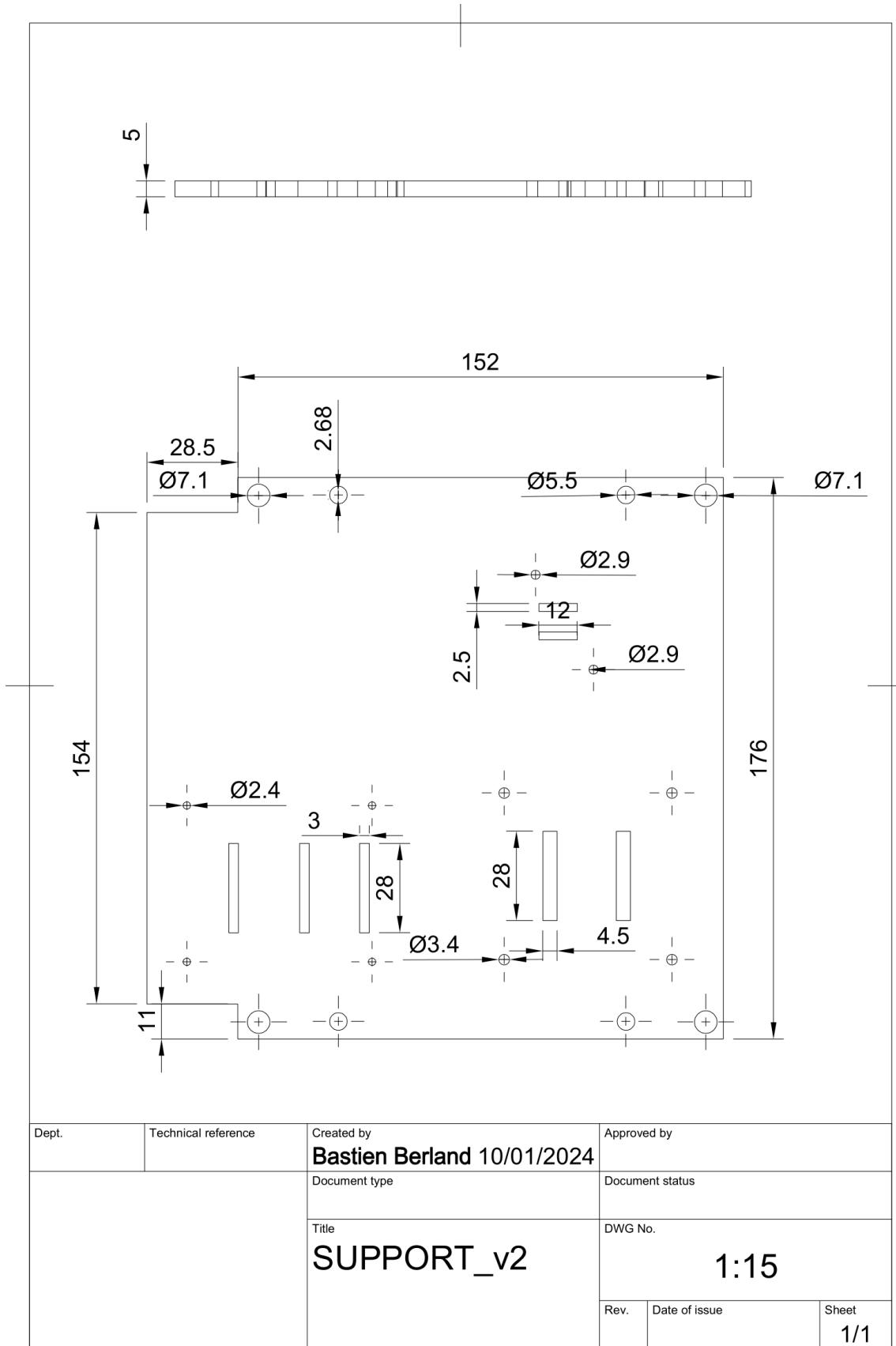
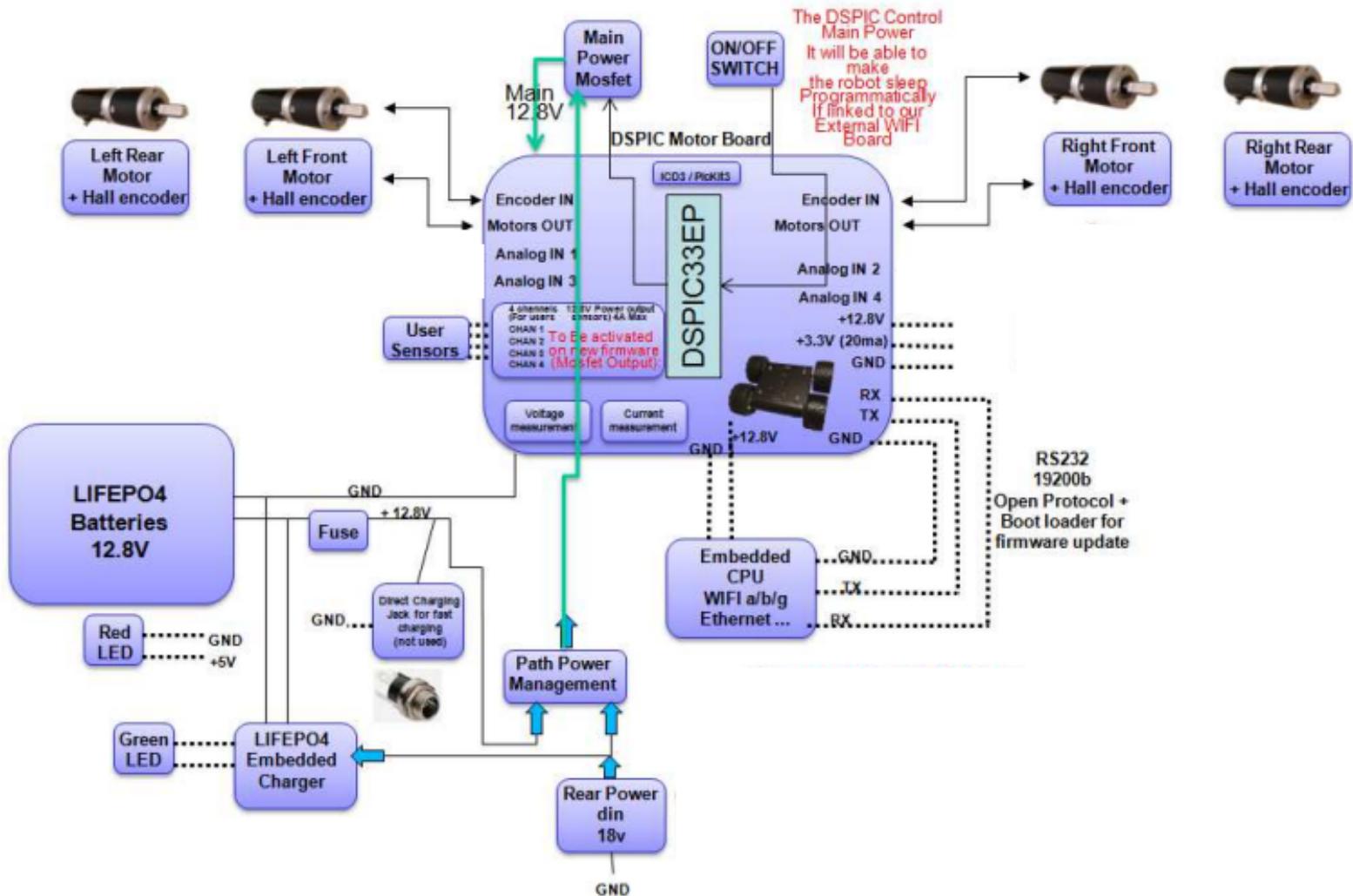


Schéma U



WifibOT LAB V4

Programme Python

```
1  from collections import deque
2  from imutils.video import VideoStream
3  import argparse
4  import cv2
5  import imutils
6  import time
7
8  fonts = cv2.FONT_HERSHEY_COMPLEX
9  WHITE = (255, 255, 255)
10
11 # construct the argument parse and parse the arguments
12
13 ap = argparse.ArgumentParser()
14 ap.add_argument("-v", "--video",
15     help="path to the (optional) video file")
16 ap.add_argument("-b", "--buffer", type=int, default=64,
17     help="max buffer size")
18 args = vars(ap.parse_args())
19
20 ColorLower = (29, 86, 6)      #red#(160,100,20)
21 ColorUpper = (64, 255, 255)    #red#(179,255,255)
22 pts = deque(maxlen=args["buffer"])
23
24 if not args.get("video", False):
25     vs = VideoStream(src=0).start() #set up usb port
26
27 else:
28     vs = cv2.VideoCapture(args["video"])
29
30 time.sleep(2.0)
```

```
31
32
33     while True:
34         # grab the current frame
35         frame = vs.read()
36         frame = frame[1] if args.get("video", False) else frame
37
38         if frame is None:
39             break
40
41
42         frame = imutils.resize(frame, width=600)
43         blurred = cv2.GaussianBlur(frame, (11, 11), 0)
44         hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
45
46
47         mask = cv2.inRange(hsv, ColorLower, ColorUpper)
48         mask = cv2.erode(mask, None, iterations=2)
49         mask = cv2.dilate(mask, None, iterations=2)
50         cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
51
52         cnts = imutils.grab_contours(cnts)
53
54         center = None
55
56         # only proceed if the ball is found
57
58         if len(cnts) > 0:
59
60             c = max(cnts, key=cv2.contourArea)
61             ((x, y), radius) = cv2.minEnclosingCircle(c)
62             M = cv2.moments(c)
63             center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
64
65             if radius > 15 :
66
67                 cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 3)
68                 cv2.circle(frame, center, 5, (0, 0, 255), -1)
69
```

Projet tutoré 2023-2024

```
70
71     if radius != 0 :
72         cv2.putText(frame, f"Distance = {round(72.24*0.98**int(radius),2)} cm", (50, 50), fonts, 1, (WHITE), 2)
73
74     pts.appendleft(center)
75
76
77     # show the frame to our screen
78     cv2.imshow("Frame", frame)
79     key = cv2.waitKey(1) & 0xFF
80     # if the 'q' key is pressed, stop the loop
81
82     if key == ord("q"):
83         break
84
85     # if we are not using a video file, stop the camera video stream
86     if not args.get("video", False):
87         vs.stop()
88     # otherwise, release the camera
89     else:
90         vs.release()
91     # close all windows
92     cv2.destroyAllWindows()
```