



PHY15 - Utilisation de vecteurs motorisés
autonomes pour la détection de foyer
d'incendie en parc de stationnement couvert

Charles Nabbout - Tristan Cheny - Sinitandjon Yaya Yeo -
Abdellah Bulaich Mehamdi - Adrien Combelles

Remerciements

Nous tenons à exprimer notre gratitude vis-à-vis de toutes les personnes qui ont contribué à la réalisation de ce PSC.

Nous remercions notre tuteur le capitaine Mathieu Giroir pour avoir proposé le sujet.

Nous remercions le capitaine Julien MARTIN pour son engagement, le temps qu'il nous a accordé afin de rendre clairs les objectifs et les principaux axes du projet et sa diligence lorsque nous avions des questions à lui poser, malgré son calendrier chargé.

Nous tenons également à remercier nos coordinateurs Alistair ROWE, Guillaume GALLOT et François OZANAM du département de physique, pour les aides, les conseils, les encouragements et la disponibilité qu'ils nous ont accordés.

Nous remercions enfin Gareth PATERSON, notre contact au Drah-X, pour la formation nécessaire à la construction du chariot.

Executive summary

L'objectif principal de ce PSC est de concevoir un **dispositif autonome aidant les pompiers à détecter le foyer d'un incendie en parking**, où la vue est bloquée par une fumée opaque, l'air est brûlant et parfois irrespirable, et où la structure menace de s'effondrer à tout moment. Cette année, nous avons travaillé sur un système de cartographie en temps réel.

Le système de cartographie est basé sur une carte d'émission et réception d'ultrasons HC-SR04. Le choix des **ultrasons** s'est fait tôt dans l'année, sur plusieurs critères. À l'instar d'un radar, le capteur balaye son environnement et mesure le temps de retour de l'écho, ce qui lui permet de calculer les distances des obstacles qui lui font face.

Après une réflexion minutieuse sur le choix de la **structure de données** à employer, choix primordial car il conditionne l'efficacité de toute la suite du projet (problème d'efficacité de l'algorithme, ...), le système obtenu fonctionne et est fiable jusqu'à une distance de 2 mètres. La structure choisie nous permet d'effectuer un **filtrage** des données sommaire, mais efficace. Le tout est affiché dans une fenêtre interactive.

Le robot doit être capable de connaître sa **position absolue** à chaque instant, afin de réaliser une carte utilisable par les pompiers. Nous avons étudié la faisabilité d'utilisation d'une **centrale inertuelle**. C'est le moyen de localisation le plus fiable parmi d'autres, comme le positionnement GPS (difficulté de réception dans un parking souterrain), ou le repérage en posant des balises (problèmes logistiques).

Sans mettre en œuvre le dispositif de la centrale inertuelle qui nécessite le robot final, nous avons construit un chariot se déplaçant à vitesse constante, afin de tester notre algorithme de cartographie.

Au vu de la quantité importante de données à traiter et de la puissance de calcul limitée de la carte Arduino, il faut traiter les données dans une machine externe. Il est important d'établir une bonne communication entre la carte Arduino et la machine, pour ce faire nous avons étudié la **communication par radiophonie**.

Le module d'émission et de réception radio basé sur la puce NRF24L01 est simple à utiliser avec une portée qui en théorie est à 1000 mètres mais qui peut se réduire à 50 mètres dans notre cas. Le système LoRa "Long Rang" a une portée plus efficace.

Table des matières

1	Introduction	5
2	Objectifs et organisation du projet	6
2.1	Présentation du sujet et objectifs	6
2.2	Organisation du travail	6
3	Effet de l'environnement sur l'acquisition	8
3.1	Acoustique de la combustion	8
3.2	Mesures	9
3.3	Simulation de la propagation du son	9
4	Principe d'acquisition et capteur	11
4.1	Dispositif choisi	11
4.2	Réalisation	12
4.3	Langage utilisé	13
4.4	Acquisition	14
5	Traitement des données	16
5.1	Choix des structures de données	16
5.2	Principe de l'algorithme	17
5.3	Affichage des résultats	19
6	Test en déplacement	20
6.1	Construction d'un chariot	20
6.2	Capteur et chariot	21
7	Étude d'un système permettant la détermination de la position absolue du robot	26
7.1	Motivation	26
7.2	Comparaison de différentes méthodes	26
7.3	Étude de la fiabilité d'une centrale inertielle	28
8	Communication et transmission des données par radiophonie	31
8.1	Motivation	31
8.2	nRF24L01	31
8.3	LoRa et LoRaWAN	32
9	Conclusion	34
	Bibliographie	35

1 Introduction

Ce projet, proposé par la brigade des sapeurs-pompiers de Paris (BSPP), est mené sur plusieurs années par plusieurs groupes de PSC consécutifs, chacun pouvant se concentrer sur certains aspects du problème. Le but initial est de réaliser un robot capable de localiser de manière autonome le foyer d'un incendie en parc de stationnement couvert tout en pouvant transmettre aux pompiers une cartographie de l'environnement dans lequel il progresse. Au regard des centres d'intérêts et des compétences des membres de groupe, c'est sur ce dernier aspect du projet que nous avons convenu avec nos interlocuteurs de la BSPP de travailler.

Nous avons donc décidé de nous concentrer sur l'élaboration d'un système de cartographie, permettant à un appareil évoluant dans un milieu typique d'un incendie en parc de stationnement couvert, de réaliser et de communiquer une carte rudimentaire de son environnement.

Tout au long du projet, nous avons décrit et défini de plus en plus précisément le système que nous voulions élaborer d'ici la fin de l'année, et bien avancé dans sa réalisation. Nous avons choisi le type de signal et de capteurs, travaillé sur des algorithmes nécessaires au fonctionnement de notre système et écrit des programmes pour les implémenter. En parallèle, nous avons travaillé sur l'impact de l'environnement de l'appareil, pour essayer de prévoir les causes d'éventuels dysfonctionnements et de justifier le choix de certains composants dans la réalisation du système.

Cette année de projet ne s'est bien sûr pas déroulée sans incident et sans remise en question. Nous avons notamment été amené à redéfinir ou du moins préciser nos objectifs conjointement avec les coordinateurs, lors de la réunion de cadrage initiale mais également à mi-parcours. Nous n'avons par ailleurs malheureusement pas pu réaliser de test en conditions réelles avec les pompiers, ce qui aurait indiscutablement porté le projet plus loin. Notre conception même du projet a évolué, et nous considérons notamment pour la suite un robot contrôlé par les pompiers et pas forcément autonome.

Dans ce rapport, nous rendons compte de l'avancée du projet au terme de cette année de travail, en décrivant les travaux réalisés, présentant et critiquant les résultats obtenus et enfin, en ouvrant sur la suite à donner au projet et en proposant des pistes aux groupes de PSC qui seront éventuellement amenés à prendre le relais dans les années qui viennent.

2 Objectifs et organisation du projet

2.1 Présentation du sujet et objectifs

Le sujet initial, proposé par la BSPP, avait pour objectif la réalisation d'un vecteur motorisé pour la détection d'un foyer d'incendie en parc de stationnement couvert qui permettrait de limiter l'exposition des pompiers. En effet, ce sont eux aujourd'hui qui font ce travail de reconnaissance, en se mettant en grand danger car dans un parking couvert en feu l'air confiné est vicié et le risque d'effondrement de la structure est rapidement élevé. Le coeur du projet semblait donc se situer naturellement dans l'étude des paramètres physiques mesurables indiquant la proximité du foyer, et dans la réalisation ultérieure d'un appareil utilisant ces paramètres pour localiser le foyer.

Finalement, il nous est apparu lors de la présentation du sujet par notre interlocuteur à la BSPP, le CNE Julien Martin, que l'enjeu était plus vaste et que la capacité du robot à fournir une cartographie de son environnement était aussi un point important, puisque c'est dans cet environnement que les pompiers évoluent ensuite pour aller éteindre le foyer. C'est donc sur ce dernier aspect du projet, répondant davantage aux centres d'intérêts de la majorité des membres du groupe car tourné vers l'informatique, que nous avons choisi de travailler.

Initialement, nous avions défini comme objectifs de concevoir un système de cartographie qui trace la carte en temps réel, de trouver un moyen de transmission entre le robot et les pompiers, et s'il restait du temps, de commencer à travailler sur la partie détection du foyer. Ces objectifs, trop ambitieux, ont été redéfinis et précisés à la réunion de cadrage avec les coordinateurs: d'abord choisir quel signal utiliser pour le capteur et obtenir un dispositif traçant une carte en restant immobile. Une fois ces objectifs atteints, nous avons convenu avec les coordinateurs de travailler sur le temps restant à réaliser un dispositif traçant une carte en déplacement, à mettre en place une démonstration, et à commencer à étudier la transmission du signal et la possibilité d'utiliser une centrale inertuelle pour connaître le déplacement du robot (nécessaire à notre algorithme comme nous le verrons), objectifs que nous avons remplis.

2.2 Organisation du travail

La division du travail s'est faite en général en fonction des centres d'intérêts. Initialement, Charles, Adrien, Yaya et Abdellah se sont concentrés sur la

partie conception de l'algorithme, Tristan et Charles sur la partie choix du signal et de la méthode de mesure, et Tristan était le contact de notre interlocuteur à la BSPP. Dans la deuxième partie de l'année, Charles a travaillé seul sur le code, Abdellah et Yaya ont étudié la méthode de transmission du signal, Adrien l'utilisation de la centrale inertIELLE et Tristan s'est concentré sur la réalisation d'un chariot motorisé pour la démonstration finale en mouvement.

3 Effet de l'environnement sur l'acquisition

Nous avons décidé d'utiliser des signaux ultrasonores pour réaliser la cartographie de l'environnement de l'incendie. En effet, il s'agit du type d'onde communément utilisé pour ce genre d'applications. Or, nous savons que la combustion peut être à l'origine d'ondes acoustiques. Il n'est donc pas exclu que ces ondes interfèrent avec notre signal, compromettant ainsi les résultats. Il convient donc d'anticiper cette possibilité en déterminant si les fréquences que peut produire la combustion sont proches des fréquences ultrasoniques utilisées par l'appareil.

Un autre aspect physique de l'environnement qui pourrait impacter le bon fonctionnement de l'appareil est la probable forte inhomogénéité de l'air ambiant, notamment en terme de température. Celle-ci résulte en d'importantes variations d'indice de réfraction n pouvant dévier la trajectoire des signaux ultrasoniques, faussant ainsi les mesures de l'appareil. Il convient également d'étudier ce phénomène afin d'anticiper ce risque.

3.1 Acoustique de la combustion

La production de son par une combustion est notamment liée aux variations de la production volumique des produits de la combustion. Cette variation comprime ou dilate l'air ambiant, occasionnant la propagation d'une onde de pression. **Les fréquences sonores habituellement associées au bruit de combustion dans le cas d'une flamme turbulente usuelle sont majoritairement comprises entre 270 et 700Hz et ne dépassent pas 3kHz, soit bien inférieures aux 40kHz du signal ultrasonore que nous avons choisi d'utiliser.**

Cependant, **d'autres phénomènes peuvent également produire du son.**

Lorsqu'une flamme est fortement déformée par la turbulence, différents fronts de flamme sont plus susceptibles de se toucher, résultant en un temps très rapide en l'annihilation des fronts de flamme et la libération de poches de gaz en combustion. Ce processus est à l'origine d'une émission de bruit courte, intense et à une fréquence pouvant être de l'ordre de 10kHz, soit le même ordre de grandeur que notre signal. Ce phénomène n'apparaît toutefois que dans le cas de flammes très turbulentes, dans des configurations précises, et le risque d'interférences reste donc faible. Nous voulions tout de même réaliser des mesures afin d'en être sûr.

3.2 Mesures

Nous avions prévu d'effectuer les mesures du signal acoustique émis par un incendie, à l'aide d'un émetteur-récepteur ultrason, afin de définitivement écarter le risque d'interférences et valider par l'expérience le choix d'un signal ultrasonore de fréquence $40kHz$ pour notre dispositif.

Malheureusement, malgré nos demandes et l'implication de notre interlocuteur, nous n'avons pas pu assister à un brûlage avec la BSPP et donc réaliser ces mesures.

3.3 Simulation de la propagation du son

De manière à tester la propagation des ondes dans un air avec un gradient d'indice non nul, nous avons conçu un programme permettant la simulation de cette propagation en 2D.

Nous avons utilisé la formule de propagation des rayons dans un milieu non homogène :

$$\frac{dn \frac{\partial \vec{r}}{\partial t}}{ds} = \overrightarrow{\text{grad}} n$$

Le but du programme est de simuler la propagation de rayons lumineux partant d'une source émettant un rayon entre $-\theta$ et θ . Pour cela, le programme prend en entrée la fonction $n(x, z)$ donnant la valeur de l'indice de réfraction du milieu, et intègre l'équation précédente pour chaque rayon.

Plutôt que de résoudre numériquement l'équation précédente telle quelle, on la multiplie par n et on procède au changement de variable $d\tau = \frac{ds}{n(s)}$. On résout donc l'équation suivante, plus simple :

$$\frac{d^2 \vec{r}}{d\tau^2} = \overrightarrow{\text{grad}}(n^2)$$

En utilisant le programme `deviation.py` présenté en annexe, on montre deux simulations dans les cas $n(x, z) = 1 - |z - 1|$ et $n(x, z) = 1 + |z - 1|$.

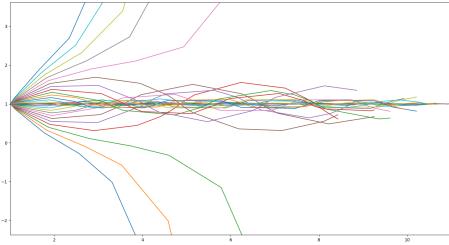


Figure 1: Guide d'onde

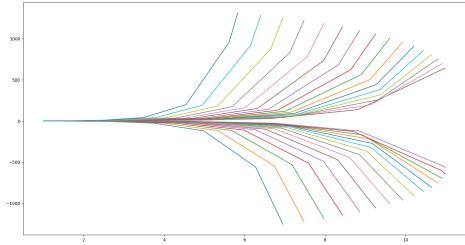


Figure 2: Zone "blanche"

Cet algorithme nous permettra d'évaluer avec les mesures qui seront prises s'il y a un risque que les ondes émises par notre radar soient déviées. Le code est disponible en annexe.

Selon notre interlocuteur à la BSPP, dans tout le parking, la température est constante égale à 60-70°C, dans un rayon de 30m du véhicule, elle est de 400°C, et au niveau du véhicule, elle est de 1200°C. Nous prenons une modélisation grossière où la température, normalement due au rayonnement, suit une loi linéaire $T(x) = 400 + 26.7x$. Voici le résultat de notre simulation :

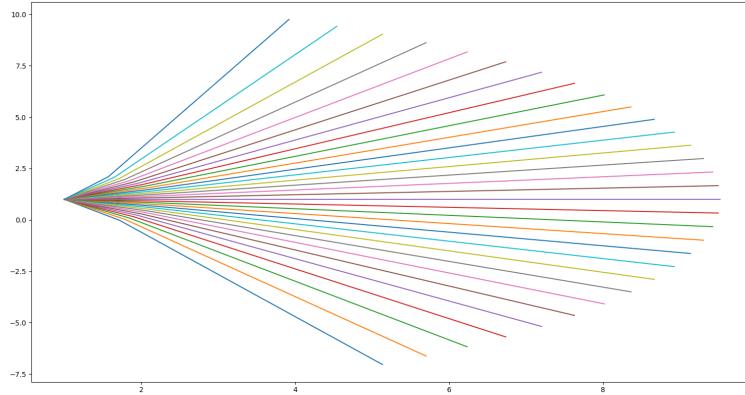


Figure 3: Réfraction des rayons

On observe que les rayons sont peu déviés, donc, **notre robot ne devrait pas avoir de problèmes** même s'il s'approche de la source des flammes, et

quitte l'environnement privilégié où la température est quasi-uniforme égale à 60-70°C.

4 Principe d'acquisition et capteur

4.1 Dispositif choisi

Le dispositif choisi afin de permettre au robot de procéder à la reconnaissance de son environnement est basé sur le principe d'un radar. Suivant le conseil des coordinateurs qui nous avaient très vivement recommandé de partir d'une base fonctionnelle, nous nous sommes inspirés d'un modèle qui fonctionnait déjà. Nous nous sommes appuyés sur ce site [1].

Il s'agit d'une **carte d'émission/acquisition d'ultrasons HC-SR04**, qui émet un signal ultrason dans la direction lui faisant face, et qui renvoie à la carte la mesure du temps qu'a mis l'écho à revenir. On a donc accès à la distance du point sur lequel l'onde sonore s'est réfléchie, puisque la vitesse du son est directement reliée à la température de l'air (et son humidité, mais le feu est plutôt sec). Dans nos tests, c'est à dire entre 10°C et 40°C, la vitesse du son dans l'air reste entre 337 m/s et 350 m/s, d'où la valeur prise constante égale à 340m/s. Il est tout à fait envisageable d'avoir un algorithme qui prend en compte la température de l'air, reçue via un des capteurs utilisés l'an dernier, pour modifier la vitesse du son utilisée dans le calcul de la distance, de manière autonome.

Cette carte HC-SR04 est montée sur un **moteur**, dont nous contrôlons la rotation au degré près. Le tout est branché à une carte Arduino, qui nous permet de faire l'interface entre le capteur/moteur et l'ordinateur.

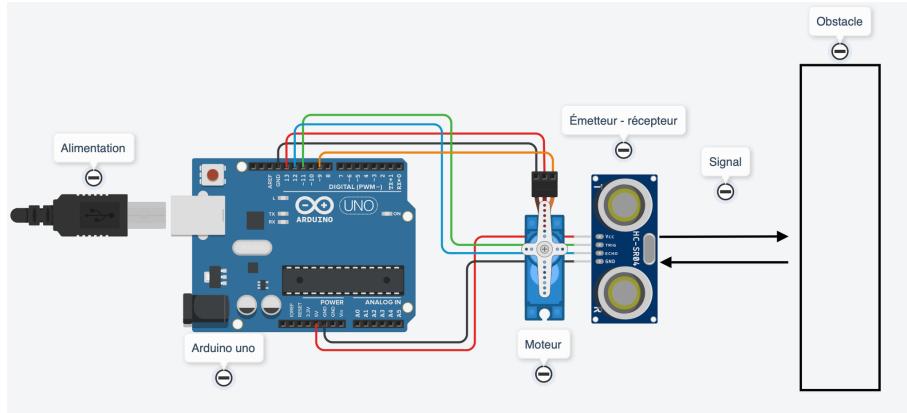


Figure 4: Représentation du dispositif

Le principe est le suivant : le moteur tourne d'un degré à chaque fois, et pour chaque degré, on lance un signal ultrason afin de déterminer la distance de l'objet qui fait directement face au capteur. Cette information est ensuite transmise à l'ordinateur grâce à l'Arduino. Le moteur fait ainsi des allers-retours, et détermine la position des obstacles qui l'entourent.

4.2 Réalisation

Nous avons donc réalisé une implémentation de ce dispositif. Bien sûr, l'objectif final du projet est d'avoir un dispositif compact capable de tenir sur un véhicule de très petite taille. Nous avons donc **limité le nombre de composants** différents : une carte Arduino, un moteur, un capteur. Nous les avons placés sur une plaque mobile, dont le seul câble sortant est le câble de liaison ordinateur/arduino.

Les différents éléments sont espacés sur la plaque pour des questions de lisibilité, mais nous avons pris soin de faire en sorte qu'ils puissent être rapprochés si besoin est. Dans l'absolu, il est même possible de s'affranchir de la grille de connexion blanche en connectant les câbles directement entre eux. On perdrait cependant en lisibilité, et donc en vitesse de dépannage du système dans une phase aussi avancée du développement. Notons donc que sa présence est effectivement superflue à terme.

Actuellement, le dispositif nous convient quant à sa praticité : il permet en effet des débogages et dépannages faciles, comme visible sur la figure 5. Le principe de mettre le système sur une plaque indépendante est de pouvoir

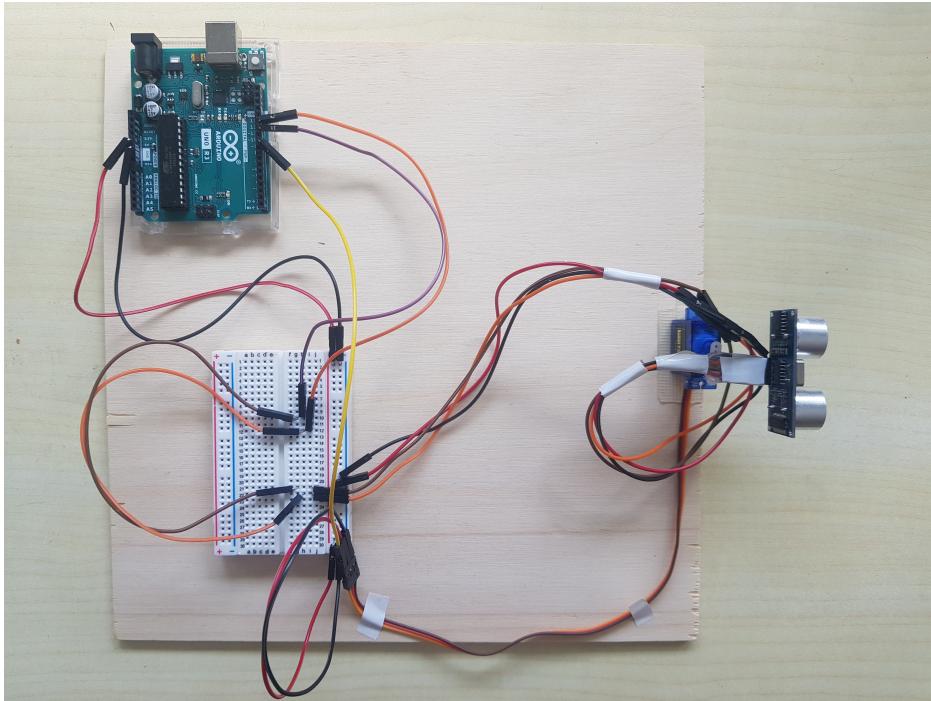


Figure 5: Capteur

le placer sur le chariot et tester le capteur en mouvement.

4.3 Langage utilisé

L’acquisition des données, et plus généralement l’ensemble du code, est écrit en **C++**. En effet, le langage est rapide, et donc particulièrement adapté à du traitement d’un grand nombre de données. De plus, il est peu gourmand en ressources, ce qui permet d’avoir de la ”marge” si jamais on doit rajouter des fonctionnalités. La possibilité de coder en séparant les fichiers permet plus de modularité et lisibilité, ce qui est important sur un projet se déroulant sur plusieurs années. De plus, la possibilité de faire de l’orienté objet avec ce langage nous a également été utile lors du développement du projet. Enfin, le langage dispose de deux librairies gratuites que nous avons exploité.

D’une part, nous disposons de la librairie `termios.h` [4] qui permet d’**acquérir les données de la carte Arduino** en lisant les données provenant du port série USB. Nous étions déjà familiers avec cette librairie, contrairement aux librairies d’acquisition d’autres langages (comme Python). Ceci

nous a donc, pour des contraintes de temps, poussé à coder en C++. Néanmoins, plusieurs problèmes ont émané de l'utilisation de cette librairie, notamment lors de la synchronisation de la lecture/écriture du port entre l'Arduino et l'ordinateur, problème qui aura mis plus d'un mois pour être résolu.

Ce sont des problèmes qui n'apparaissent pas avec la librairie `Serial` en Python, mais le langage est beaucoup plus lent (d'un facteur de l'ordre de **100 fois !**), ce qui n'est pas envisageable (sans compter que personne dans le groupe ne savait gérer un affichage de données en temps réel en Python).

D'autre part, nous avons pu utiliser la librairie `SDL2.h` qui est une librairie permettant la gestion de fenêtre, lecture de l'entrée clavier, affichage de pixels, etc. Utilisée initialement à des fins de débogage, afin simplement d'avoir un affichage des résultats d'acquisition en cours de développement du prototype, nous avons finalement décidé d'utiliser de manière définitive cette librairie afin d'**afficher l'ensemble des points relevés en temps réel**.

Cette librairie est de plus relativement portable, dans le sens où il est plus facile d'exécuter cette application sur différents systèmes d'exploitation sans avoir à tout recoder.

De plus, le langage natif de l'Arduino est extrêmement proche du C++. Nous trouvons donc encore un gain de temps à ce niveau, afin de coder le fonctionnement du système.

4.4 Acquisition

Dans son implémentation actuelle, le moteur fait des allers-retours d'une amplitude de 200°, prenant une mesure par degré. L'Arduino renvoie deux entiers : d'abord l'angle (entier > 0), puis la distance (comprise entre 0 et 300 cm), i.e. les coordonnées polaires du point mesuré, dans le référentiel dont le centre est le capteur. Le code est disponible sur GitHub.

Les points ainsi relevés par l'Arduino sont aussitôt envoyés à l'ordinateur sous un format défini par nous-même. Il s'agit de paquets de 10 bits (pas de padding effectué). Ces points sont ensuite traités par l'algorithme sur l'ordinateur, et stocké dans une structure de donnée appropriée à notre problème.

Enfin, une mesure est réalisée tous les 5 à 6 ms. Ce choix de temps a été déterminé heuristiquement, en lien avec la qualité du capteur. En effet, quand le temps de mesure devient trop faible, on récupère des données trop

bruitées que le programme n'arrive plus à filtrer. De plus, ce temps de mesure permet d'offrir suffisamment d'information pour créer une "carte" sommaire en un temps relativement court (un demi-balayage ne dure que 1.2 s). Nous verrons plus tard que ceci pose néanmoins quelques problèmes.

5 Traitement des données

5.1 Choix des structures de données

Le choix des structures de données utilisées dans le code est une étape cruciale dans l'implémentation de l'algorithme. Au début du projet, nous pensions que le programme de traitement des données serait exécuté directement par le robot, ce qui induit des contraintes matérielles importantes concernant la mémoire et la puissance de la puce de la carte Arduino. En effet, le choix d'une mauvaise structure de données peut aboutir à un programme trop lent, ce qui nécessiterait de ré-optimiser le code. D'expérience, c'est toujours une opération pénible et chronophage.

En fin de compte, nous avons observé qu'il était bien plus simple pour le robot de simplement envoyer les mesures de points à une tablette/ordinateur détenue par un pompier placé en sûreté. C'est donc cette tablette qui est en charge de l'affichage et du traitement des données. Ceci permettra dans le futur d'économiser de la batterie, puisque la carte demande moins de puissance pour effectuer ses opérations.

Vers la période de Novembre, nous avons décidé de passer un temps substantiel à réfléchir sur la meilleure structure de données à adopter pour nous et surtout pour nos successeurs. Nous avons décidé d'opter pour une structure plus simple, mais surtout plus souple, après avoir réfléchi à l'ensemble des contraintes imposées par le robot et son environnement, et l'ensemble des caractéristiques dont l'algorithme aurait besoin et que nous pouvions prévoir.

La structure a été choisie pour des raisons algorithmiques, au vu des contraintes auxquelles notre algorithme est soumis, ainsi que pour sa simplicité. C'est finalement une **liste doublement chaînée** qui est utilisée. C'est une structure élémentaire, et il est facile de modifier le code afin de supporter une structure plus complexe. C'est d'ailleurs pour cela que la liste des points est **encapsulée** dans un **struct list_Points**.

Les contraintes sont :

- nécessité de parcourir l'intégralité des points au moment de l'affichage/traitement des données
- nécessité de supprimer des éléments arbitrairement placés dans la liste
- nécessité d'ajouter des éléments à la fin de la liste

La structure utilisée exécute ces deux dernières opérations en temps constant. De plus, le parcours de la liste est facilité par l'usage d'**itérateurs** : c'est

la structure idéale.

5.2 Principe de l'algorithme

On stocke la position relative du capteur par rapport à sa position initiale (au moment où l'on lance l'acquisition) dans la classe `Position`. Les coordonnées polaires fournies par la carte Arduino (angle et distance) sont converties en coordonnées cartésiennes, avec pour origine la position initiale du capteur. L'affichage des points est discrétilisé au centimètre, i.e. 1 pixel fait 1 cm de côté.

Le principe est le suivant : aussitôt qu'un point est reçu par le programme sur l'ordinateur, il subit un premier filtrage élémentaire. Ce filtrage consiste en un moyennage des valeurs trop discontinues : on prend trois points successifs dans la liste, dont les mesures des distances au capteur sont m_1 , m_2 , et m_3 . Si $m_1 \approx m_3$ et m_2 est éloigné de m_1 et m_3 , on modifie la valeur de m_2 en la moyenne de la position de ses voisins : $m_2 = \frac{m_1+m_3}{2}$. Ce premier filtrage permet de normaliser les données d'entrée, ce qui simplifie le travail du second filtre.

On applique ensuite un second type de filtrage. Dans la suite, la constante `NB_FILTRE` est choisie égale à 5. Pour chaque point, on teste si parmi les `NB_FILTRE - 1` points précédents et les `NB_FILTRE - 1` points suivants, il existe au moins un point présent à une distance plus faible que `seuil`. Nous avons choisi `seuil` égal à 5 cm, de sorte qu'un point isolé de ses $2 * \text{NB_FILTRE} - 1$ voisins d'une distance de plus de $\text{seuil} = 5$ cm est automatiquement supprimé. On conçoit un cas test ci-dessous :



Figure 6: Test **filtre2**

Le choix d'avoir pris **seuil** égal à 5 cm correspond à l'heuristique suivante : 2 points séparés d'un angle de 1° situés sur le même arc de cercle de rayon 2m sont séparés d'une distance d'environ 3 cm. Ainsi, si un point est à 5 cm de ses 9 voisins, alors, il est probablement isolé dans la vraie vie, et on peut voir les résultats de ce filtrage sur les images suivantes (à gauche, pas de filtrage) :

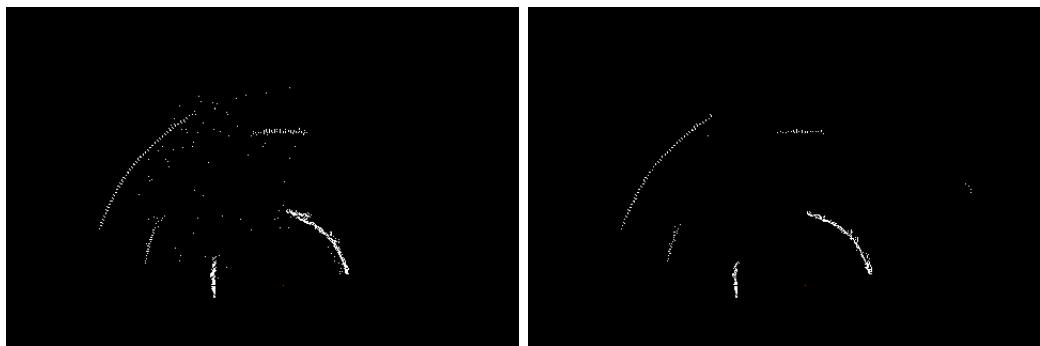


Figure 7: **seuil** = 100

Figure 8: **seuil** = 5

Le choix de l'hyper-paramètre **seuil** est donc purement heuristique. Nous avons testé des valeurs jusqu'à 1cm. En-dessous, le filtre masque trop de points, et on ne perçoit plus rien sur la carte.

5.3 Affichage des résultats

Nous ne rentrerons pas dans les détails de l'implémentation de la carte. Les points sont affichés à l'écran avec la correspondance : 1 pixel fait 1 cm*1cm. Le point rouge indique la position du capteur. Les points blancs indiquent les points relevés par le capteur. On peut se déplacer sur la carte en utilisant les flèches directionnelles sur le clavier. Nous ne l'avons pas implémenté, mais il est possible via SDL de coder le déplacement de la carte via la souris.

SDL étant tout particulièrement optimisé pour l'affichage de petites images (communément appelés blits), il est enfin possible de mettre dans un coin une image de boussole indiquant dans quelle direction le feu a été détecté, via les outils du PSC de l'an dernier.

6 Test en déplacement

6.1 Construction d'un chariot

Nous avons donc conçu un algorithme permettant de tracer la carte en temps réel, qui fonctionne en adaptant la carte obtenue à chaque mesure, en tenant compte du déplacement absolu du robot. A ce stade, le robot ne dispose pas de moyen de connaître sa position absolue. Afin de tester quand même l'algorithme, nous avons décidé de placer le dispositif de mesure sur un chariot se déplaçant de manière rectiligne à une vitesse constante connue, et **fournie à l'algorithme**.

Le chariot motorisé, faisant donc office de robot, est constitué des composants suivants :

- une plaque de plexiglas (coupée par *laser cutting* au DrahiX)
- deux roues arrières motorisées (référence *Gebildet 4pcs DC3V-12V DC*)
- deux roues avants libres
- piles alimentant les moteurs (l'alimentation peut également se faire par l'Arduino)
- un interrupteur marche/arrêt

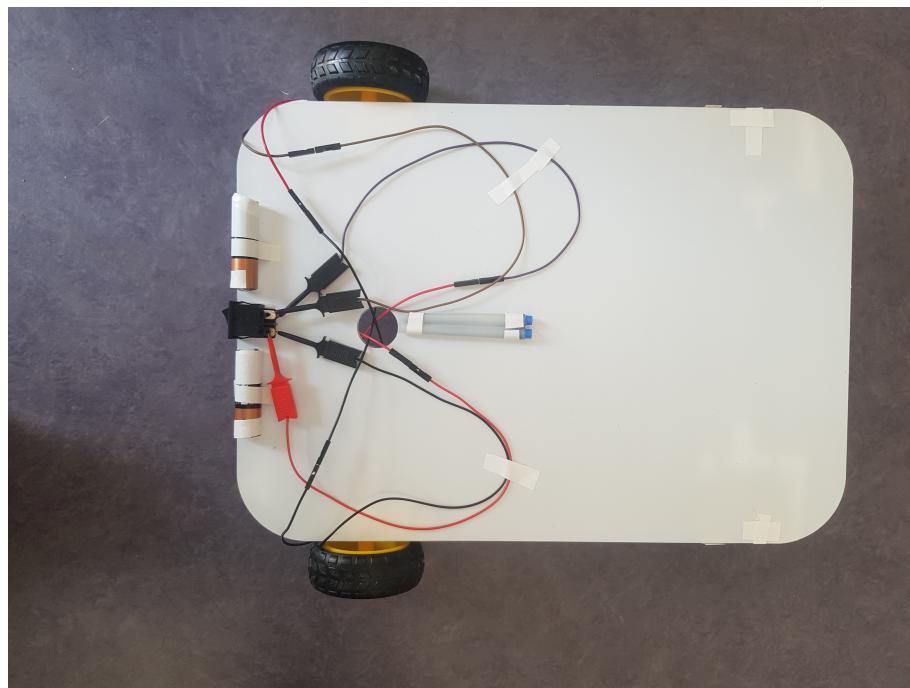


Figure 9: Chariot conçu

6.2 Capteur et chariot

On place le détecteur sur le chariot, et afin d'assurer que les moteurs tournent plus vite, on les alimente via la carte Arduino (3,5V contre 1,5V). On pallie ainsi à des problèmes de faux contacts entre câbles et piles. On obtient le système ci-dessous :

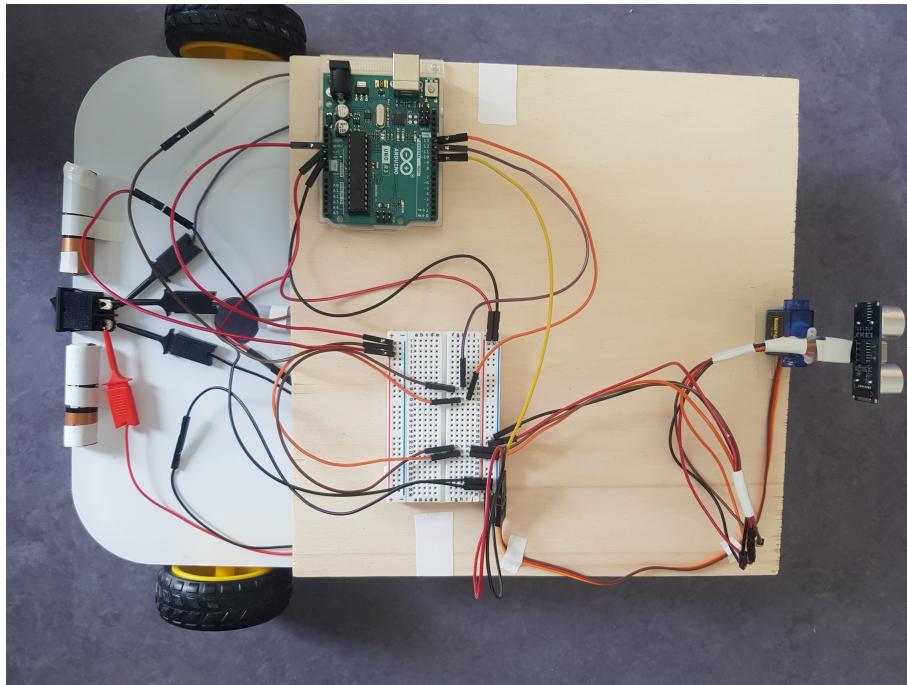


Figure 10: Chariot avec capteur positionné

Le chariot est lesté afin que les roues ne vrillent pas. On mesure la vitesse du chariot, qui est de 26cm/s. On rentre cette donnée dans l'algorithme, qui va modifier la variable "Position", associée à la position du robot tous les 25ms (toutes les 5 mesures) via la fonction `automove`. On peut ensuite placer le robot dans un petit parcours afin de tester ce que le robot réussit à percevoir et transmettre.

Nous proposons le circuit suivant, les obstacles sont entourés et numérotés en jaune (11) :

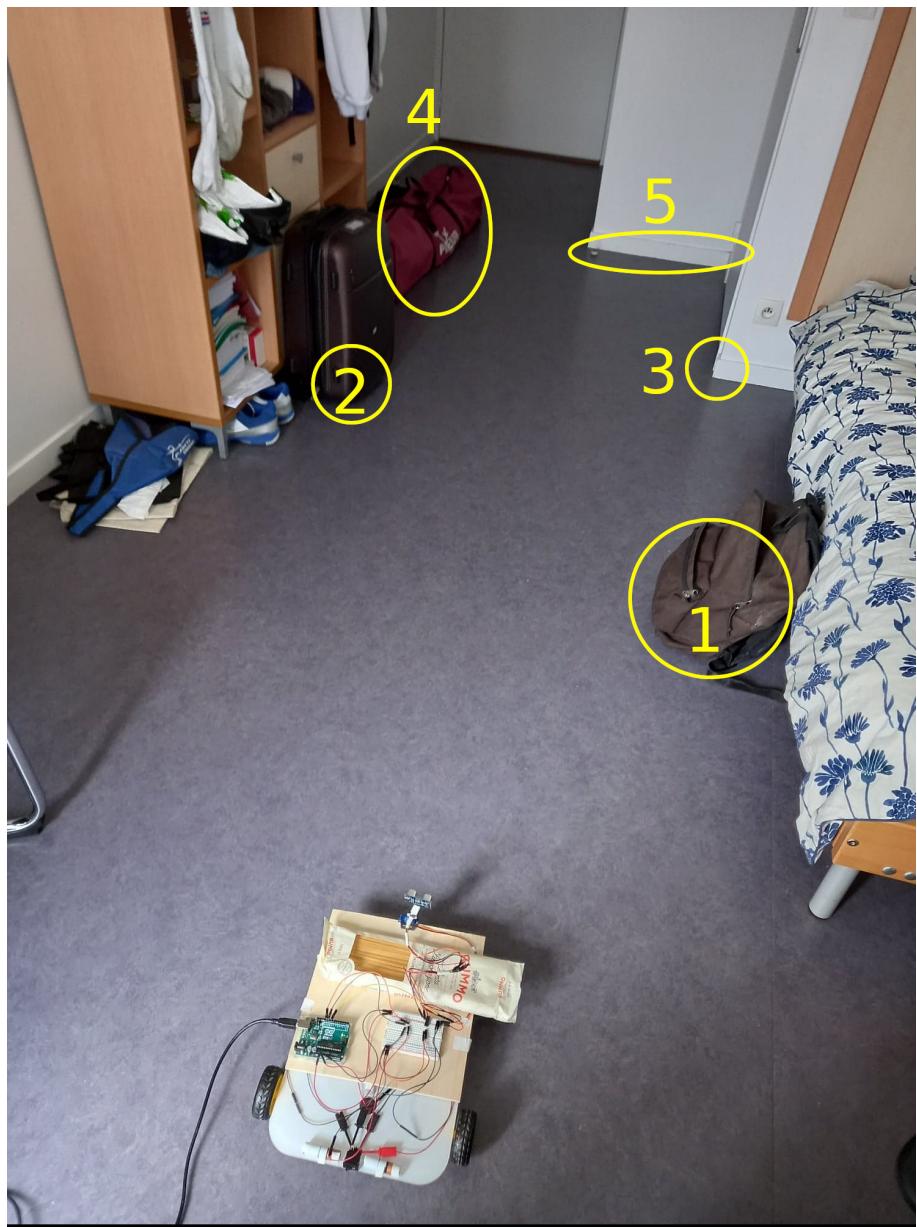


Figure 11: Parcours du robot

Voici la carte obtenue, on y reconnaît les différents éléments annotés sur l'image précédente :

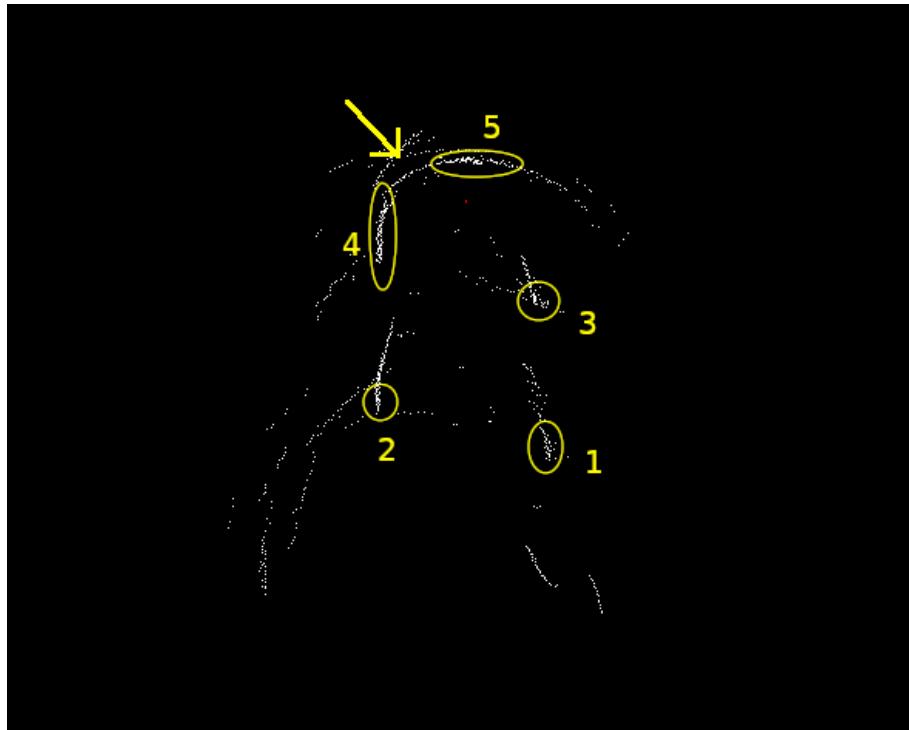


Figure 12: Scan effectué par le robot

On observe cependant, à l'endroit indiqué par la flèche un léger défaut: le couloir pourtant ouvert apparaît fermé sur le scan effectué par le robot (12). Nous avons déterminé l'origine de ce problème. En effet, la vitesse du chariot étant élevée devant la vitesse de balayage du moteur, nous avons dû accélérer cette vitesse de balayage en diminuant le temps pris par chaque mesure (à ce stade, le temps de chaque mesure est passé de 10ms à 5-6ms).

Le problème est donc que, pour des points situés à plus d'1.3 mètres, **le moteur tourne avant que l'écho des points précédents ne parvienne au robot** (calcul élémentaire : $340\text{m/s} * 5\text{ms} = 1.7\text{m}$). Une observation pendant le scan effectué par le robot a montré que ces points "fictifs" correspondent à une mesure décalée du sac rouge (4) et du mur (5). Ces points apparaissent quand le robot est à une distance d'environ 1.3m, et n'étaient plus relevés ensuite quand le robot se rapproche des obstacles.

Nous ne pouvions pas simplement diminuer la vitesse de balayage, car dans ce cas, le robot avance trop vite et la carte n'est pas suffisamment détaillée. Nous sommes confrontés à deux choix :

- diminuer la vitesse du robot à environ 15cm/s, (et ensuite la vitesse de balayage à 10ms par mesure)
Cette option est envisageable, mais, sachant que le robot doit trouver le foyer en moins de 20min, il ne pourrait parcourir plus que 180m contre 312m actuellement ;
- diminuer la portée du robot à 1.3m
Cette option diminue l'efficacité avec laquelle le pilote du robot pourra prévoir les éventuels cul-de-sac

Le code est disponible sur GitHub.

7 Étude d'un système permettant la détermination de la position absolue du robot

Les deux dernières parties représentent le travail investi par notre groupe afin de résoudre deux problèmes annexes qui se sont posés lors du développement du projet. Il nous avait été déconseillé par les coordinateurs de travailler sur ces parties. Nous présentons néanmoins le travail fourni sur ces problèmes, ainsi que les pistes que nous aurions nous-même exploitées si nous avions eu le temps.

7.1 Motivation

Comme on l'a vu, nous avons besoin de trouver un moyen de connaître la position relative du capteur à chaque mesure, afin d'obtenir une carte unique et détaillée. Il s'agit de la composante "localisation" du problème de SLAM (Simultaneous Localization And Mapping). Puisque le projet est centré sur la partie "cartographie", il nous a semblé important d'aborder cette question par une rapide étude des solutions possibles. Nous ne pouvons pas les mettre en œuvre de façon concrète, au vu des prix des différents outils, mais nous pouvons assurer qu'elles sont viables, pour une mise en place dans le futur.

7.2 Comparaison de différentes méthodes

Plusieurs méthodes peuvent être envisagées pour mesurer les déplacements du robot et donc sa localisation en temps réel sur la carte, chacune ayant ses propres avantages et inconvénients.

Cette localisation présente la particularité de devoir être autonome. En effet, la réception de signaux extérieurs tel qu'un signal GPS n'est pas garantie dans un parking de stationnement couvert, à plus forte raison si celui-ci est souterrain. Il est donc nécessaire d'avoir un système de localisation propre au robot, quitte à le coupler par la suite à un positionnement GPS quand celui-ci est possible (afin d'avoir une redondance et une meilleure précision).

Une première approche consiste à repérer le robot par rapport à une position absolue connue dans l'espace. Au cours de notre réflexion a été évoquée l'idée de poser des balises émettrices sur le trajet du robot, dans le but de le repérer par triangulation. Cette méthode semble robuste et fiable, mais elle apporte aussi de **nombreux problèmes logistiques** (récupération

des balises dans un milieu hostile, autonomie de chaque balise, capacité de transport du robot) et de mise en œuvre (nécessité d'un système de "largage", coûts induits, ...). Elle est donc peu pertinente dans le cadre de notre projet.

Une autre idée est de recalculer la position du robot à partir des mesures cartographiques qu'il effectue à chaque instant : le robot identifie trois points "remarquables" parmi ceux qu'il relève, et se repère par rapport à ses trois points, qui sont fixes. Ceci est inspiré de ce que font les bateaux dans la Marine, se repérant avec grande précision en repérant 3 points remarquables sur la côte via le radar. Ceci permet de déduire le déplacement relatif du robot, simplement en connaissant l'évolution de la distance à ces trois points remarquables. L'avantage de cette méthode est qu'elle est **entièvement algorithmique**, ce qui évite la mise en place de capteurs physiques supplémentaires. Néanmoins cette méthode est peu adaptée au contexte d'un parking couvert. En effet, déterminer des points remarquables nécessite la détection de structures et de géométries particulières ; or un parking couvert se caractérise par une géométrie répétitive et lisse. Par exemple, si le robot progresse dans un couloir suffisamment long pour que les extrémités soient hors de portée de ses capteurs, il sera impossible de déterminer sa position par cette méthode puisque la carte obtenue est invariante par translation.

Les deux dernières solutions reposent sur des capteurs physiques intégrés au robot. On peut d'une part mesurer des grandeurs au niveau des roues du robot, que ce soit leur **vitesse via un tachymètre** ou la **distance parcourue via un odomètre**. Le différentiel mesuré entre les roues de gauche et les roues de droite permet de détecter les virages. On détermine ainsi la trajectoire du robot par intégration de sa vitesse à chaque instant, en tenant compte de ce différentiel. Néanmoins, ces mesures dépendent de l'écart entre les roues et de leur diamètre, et sont donc très sensibles à ces paramètres. Mais même sans avoir étudié la précision de ce système, sa fiabilité peut être mise en question. Par exemple, il est aisément d'imaginer qu'à cause de débris, ou liquides sur le sol, une des roues patine ou tourne dans le vide : la mesure sera dans ce cas erronée. La méthode de mesure de la vitesse instantanée ou de la distance parcourue par un tachymètre ou un odomètre n'est donc pas assez fiable pour être utilisée seule.

Il existe un système qui permet de calculer la position du robot sans être dépendant des roues : c'est la **centrale à inertie**. Il s'agit d'un instrument doté de six capteurs : trois gyromètres qui mesurent les trois composantes de la vitesse angulaire et trois accéléromètres qui mesurent les trois composantes de l'accélération. On peut dans notre cas envisager de se limiter à un mou-

vement plan, ce qui réduit le nombre de capteurs à deux accéléromètres et un seul gyromètre. Par intégration de l'accélération et de la vitesse angulaire instantanées, on obtient la position en temps réel du robot. L'avantage principal de cette technologie est qu'elle est **totalement indépendante des autres systèmes du robot et de son environnement**. L'inconvénient est que, du fait des intégrations multiples, la précision des mesures doit être très bonne pour obtenir des résultats fiables – en effet on intègre deux fois l'accélération, alors qu'avec le tachymètre une seule intégration est nécessaire.

Dans la suite, nous proposons une évaluation de la fiabilité d'une centrale inertielles dans le contexte de l'exploration d'un parc de stationnement couvert.

7.3 Étude de la fiabilité d'une centrale inertielles

Pour étudier la faisabilité de la mise en place d'une centrale inertielles, nous nous basons sur les fiches techniques de modèles existants. Nous avons sélectionné deux modèles produits par SBG Systems, une entreprise spécialisée dans la conception de capteurs inertIELS. Le tableau 1 récapitule les valeurs de dérive de chaque modèle [6] [8].

Modèle	Dérive du gyromètre	Dérive de l'accéléromètre
Ellipse	$7^\circ/h$	$14\mu g$
Apogee	$0.08^\circ/h$	$15\mu g$

Table 1: Valeurs de dérive données par le constructeur.

À partir de ces données, nous avons simulé un trajet typique du robot dans un parking souterrain. Le code Python est disponible en annexe. Le profil du trajet simulé est visible dans la figure 13. Les données obtenues par l'algorithme sont récapitulées dans le tableau 2. On a choisi comme trajet une boucle rectangulaire d'une longueur de 300m et une vitesse constante de 0,5m/s.

Le principe de l'algorithme est le suivant. Dans notre cas simple, en deux dimensions, on peut adopter des coordonnées polaires pour la vitesse, repérée par son module v et son angle de direction ϕ par rapport à l'axe Ox . Supposons que la centrale inertielles effectue une mesure tous les pas de temps dt et notons v_n et ϕ_n module et angle de la vitesse, calculés après n mesures.

On a donc :

$$v_{n+1} = v_n + a_n dt$$

$$\phi_{n+1} = \phi_n + \omega_n dt$$

où a_n et ω_n sont les mesures d'accélération et de vitesse angulaire au n^{eme} instant. Connaissant la vitesse, le vecteur position $r_n = x_n \vec{e}_x + y_n \vec{e}_y$ vérifie donc

$$r_{n+1} = r_n + \vec{v}_n dt = (x_n + v_n \cos \phi_n) \vec{e}_x + (y_n + v_n \sin \phi_n) \vec{e}_y$$

On procède à ces calculs à partir des listes donnant a_n et ω_n (construites à la main ici, renvoyées par les capteurs dans une situation réelle) pour obtenir la trajectoire du robot.

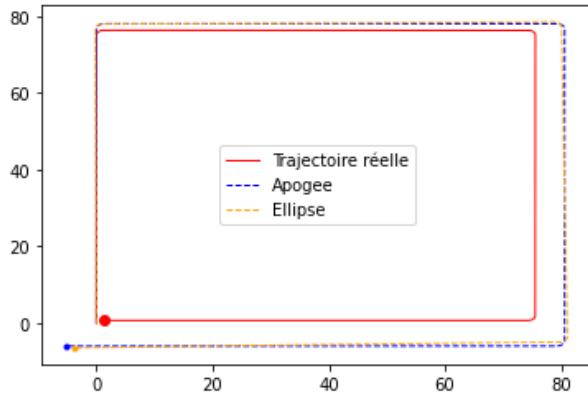


Figure 13: Déviation obtenue avec les deux modèles.

On peut tout d'abord constater que la déviation entre la trajectoire réelle et la trajectoire mesurée est d'environ 9 m, ce qui est loin d'être négligeable. Relativement à la longueur du trajet, cela représente un écart de l'ordre de 3%. L'écart angulaire est plus faible dans les deux cas : 1,2° pour le modèle Ellipse, 0,01° pour le modèle Apogee. Pour le modèle Ellipse, une correction de la dérive angulaire est envisageable grâce à une boussole intégrée au système.

Comparons ces résultats, issus d'un modèle informatique simplifié, avec un test du modèle Ellipse réalisé par SBG Systems [8]. La déviation relative qu'ils obtiennent est de 0,5%. La différence avec notre résultat peut s'expliquer par la vitesse plus élevée de leur véhicule ($6,8m/s$ en moyenne contre $0,5m/s$) et la durée d'acquisition plus faible ($365s$ contre $600s$). En relançant notre calcul avec les paramètres de leur test (durée d'acquisition

Modèle	Dérive angulaire	Déviation	Déviation relative
Ellipse	1.17°	$8.8m$	2.9%
Apogee	0.01°	$9.3m$	3.1%

Table 2: Dérives calculées pour la trajectoire typique étudiée.

de $365s$ et vitesse de $6,8m/s$), on obtient un écart relatif de 0,25%, ce qui tend à confirmer ces hypothèses.

Un écart d'une dizaine de mètres peut suffire à fausser la lecture de la carte construite par le robot, ce qui peut rendre la carte inutilisable par les pompiers. Il sera donc nécessaire de coupler la centrale inertie avec un odomètre ou un tachymètre. En effet, d'après les tests effectués par SBG Systems, le couplage leur a permis de réduire l'écart relatif de 0,5% à 0,13%. La vitesse du robot sera aussi un paramètre crucial dans la précision de la position : plus elle est élevée, moins l'erreur relative est importante. Dans notre modèle, nous avons opté pour une vitesse assez faible de $0,5m/s$, mais il s'agit d'une estimation pessimiste et on peut espérer que le robot opérationnel aura une vitesse plus élevée.

Pour conclure, l'évaluation en temps réel de la position du robot à l'aide d'une centrale inertie est **tout à fait envisageable**, néanmoins il est souhaitable de coupler cet instrument avec un odomètre et une boussole pour réduire la dérive. De plus, la dérive semble être liée à la vitesse du robot, qui devra idéalement être la plus élevée possible (en compromis avec les exigences des autres systèmes du robot). Des tests expérimentaux avec une centrale inertie dans des conditions proches du réel devront néanmoins confirmer la faisabilité de cette solution.

8 Communication et transmission des données par radiophonie

8.1 Motivation

Puisque le robot doit transmettre des données de la carte Arduino à l'ordinateur, il faut un moyen de transmission. Si l'idée d'un câble est la première qui vient à l'esprit, elle possède néanmoins le défaut majeur de ralentir le robot, potentiellement le bloquer (en faisant des noeuds, etc.). Nous avons donc étudié les différentes possibilités de transmission de données dans un environnement tel qu'un parking.

Dans notre cas, le débit minimal nécessaire est faible : on envoie un paquet de 10 bits tous les 5-6 ms, donc, le **débit minimal nécessaire est de 1,7 kbps** (ce qui est relativement faible). Après avoir observé les contraintes de portée du signal, de fumée, réflexion, murs, etc., nous avons décidé d'utiliser la radiophonie.

En se renseignant sur les cartes radios, nous avons trouvé de multiples composants variés chacun ayant leurs propres caractéristiques (débit important, portée limitée, etc.). Nous avons choisi d'étudier la carte nRF24L01 et le protocole LoRaWAN.

8.2 nRF24L01

La puce nRF24L01 est un module d'émission et réception radio, opérant sur la bande des 2,4 GHz. Elle est caractérisée par sa simplicité, sa documentation bien détaillée, son débit qui s'élève jusqu'à 2Mbps, et surtout son faible coût de 4€. La puce offre 126 canaux différents, qui ne se chevauchent pas : moins de 1 MHz de large à 250 kbps et 1 Mbps et moins de 2 MHz à 2 Mbps. Néanmoins, la sensibilité et le risque de collision dans l'air dépendent fortement du débit de la transmission : plus le débit est élevé plus la sensibilité et le risque de collision diminuent.

Parameter	Value	Unit
Minimum supply voltage	1.9	V
Maximum output power	0	dBm
Maximum data rate	2000	kbps
Supply current in TX mode @ 0dBm output power	11.3	mA
Supply current in RX mode @ 2000 kbps	12.3	mA
Temperature range	-40 to +85	°C
Sensitivity @ 1000 kbps	-85	dBm
Supply current in Power Down mode	900	nA

Figure 14: nRF24L01 Données de référence [5]

Ce module fournit certes un débit considérable et est facile à utiliser, mais la portée qu'il assure est néanmoins insatisfaisante : théoriquement de 1000m en milieu ouvert, elle est en pratique réduite à 100m, pire encore dans des milieux clos elle est de l'ordre de 50 m. Ce module est intéressant dans le sens où on peut rester suffisamment loin du robot, surtout que la distance mentionnée peut être augmentée avec l'utilisation d'une antenne, tout en ayant les informations en temps réel.

8.3 LoRa et LoRaWAN

LoRa, "Long Range", est un système de communication sans fil à longue portée, promu par l'Alliance LoRa. Ce système vise à être utilisable dans des appareils alimentés par des batteries à longue durée de vie, où la consommation d'énergie est d'une importance capitale. LoRa peut communément faire référence à deux couches distinctes : (i) une couche physique utilisant la technique de modulation radio par étalement du spectre (Chirp Spread Spectrum) et (ii) un protocole de couche LoRaWAN. Son architecture de base est la suivante : les appareils finaux utilisent LoRa en un seul saut sans fil pour communiquer avec une ou plusieurs passerelles, connectées à Internet, qui agissent comme des ponts et relaient les messages entre ces appareils finaux et un serveur de réseau central.

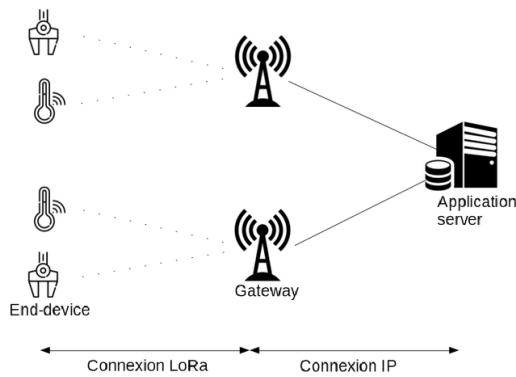


Figure 15: Architecture LoRaWAN

La modulation par étalement du spectre utilisée par LoRa consiste à répéter plusieurs fois le message transmis à des fréquences différentes. La variation de fréquence effectuée par LoRa est linéaire ce qui permet aux récepteurs d'éliminer simplement les décalages de fréquences[2].

LoRa définit le facteur d'étalement de spectre (SF - Spreading Factor) comme suit: $SF = \log_2(\frac{R_c}{R_s})$, avec R_c le débit du message transmis et R_s le débit du symbole à transmettre. Il est lié au mécanisme qui adapte la puissance d'émission et la vitesse de transmission aux conditions du réseau dans lesquelles se trouve l'objet. Il permet ainsi une meilleure gestion des ressources radio et **une optimisation de la consommation énergétique**. En outre, les facteurs d'étalement sont orthogonaux, de sorte que les signaux modulés avec des facteurs d'étalement différents et transmis sur le même canal de fréquence au même moment n'interfèrent pas les uns avec les autres.

Paramètre	Facteur d'étalement	Bande Passante	Débit Physique (bit/s)
3	SF_9	125kHz	1760
4	SF_8	125kHz	3125
5	SF_7	125kHz	5470
6	SF_6	250kHz	11000

Figure 16: Paramétrisation et débit [3]

Ce système fournit un débit de l'ordre de 5 kbps et une portée de l'ordre de 1000 m. Il garantit grâce au protocole LoRaWAN l'accès aux données collectées sur des appareils informatiques connectés et l'utilisation de multiples canaux simultanément avec la modulation par étalement du spectre. Elle réduit les interférences et augmente la précision[2].

9 Conclusion

En conclusion, nous avons tout au long du projet avancé de manière satisfaisante au regard des objectifs intermédiaires et finaux, qui avaient été revus et redéfinis avec les coordinateurs lors de la réunion de cadrage initiale et lors d'un rendez-vous pris à notre demande au milieu de l'année. Il était apparu au cours de cette première réunion que nos objectifs initiaux étaient trop ambitieux, et les coordinateurs nous avaient vivement conseillé de nous contenter dans un premier temps de réaliser un système de cartographie fonctionnant en restant immobile. C'est ce que nous avons fait, avant de nous employer dans le temps qui restait après la seconde réunion à rendre le dispositif capable de concevoir une carte en temps réel au fur et à mesure de son déplacement, et à mettre en place une démonstration.

Durant cette deuxième partie de l'année, nous nous sommes également penchés comme vu ci-dessus dans l'étude des possibilités pour la transmission du signal depuis le robot aux pompiers, et pour la connaissance de la position absolue des déplacement du robot, puisque nous avions fait le choix que l'algorithme réalisant la carte aurait accès à cette donnée. Bien que nous n'ayons pas eu le temps de mettre en place ces méthodes cette année, ces études nous permettent d'ouvrir sur la suite et de donner des idées aux groupes suivants :

1. Transmettre les données par radiophonie ;
2. Utiliser une centrale inertielle pour avoir accès aux déplacements du robot ;
3. Nous proposons de considérer l'idée d'un appareil piloté par les pompiers plutôt que totalement autonome. Les pompiers s'aideront de ces données et de la cartographie réalisée par le dispositif afin de localiser le foyer de l'incendie ;
4. Le risque de data race entre différents sous-programmes est à éviter absolument. Pour l'instant, nous savons que ceci ne pose pas de problèmes, mais mettons en garde face aux risques du *concurrent programming* dans le futur.

Bibliographie

- [1] Yug Ajmera. *Radar (SONAR) using Processing 3*. URL: <https://yainnoware.blogspot.com/p/radar-sonar-using-processing-3.html>. (consulté le 21/04/2022).
- [2] Aloës Augustin et al. “A Study of LoRa: Long Range amp; Low Power Networks for the Internet of Things”. In: *Sensors* 16.9 (Sept. 2016), p. 1466. ISSN: 1424-8220. DOI: 10.3390/s16091466. URL: <http://dx.doi.org/10.3390/s16091466>.
- [3] Lluís Casals et al. “Modeling the Energy Performance of LoRaWAN”. In: *Sensors* 17.10 (Oct. 2017), p. 2364. ISSN: 1424-8220. DOI: 10.3390/s17102364. URL: <http://dx.doi.org/10.3390/s17102364>.
- [4] Michael Kerrisk. *Page de manuel Termios*. URL: <http://manpagesfr.free.fr/man/man3/termios.3.html>. (consulté le 21/04/2022).
- [5] *Single chip 2.4 GHz Transceiver nRF24L01*. URL: https://www.sparkfun.com/datasheets/Components/nRF24L01_prelim_prod_spec_1_2.pdf.
- [6] SBG Systems. *Apogee Series Leaflet*. URL: https://www.sbg-systems.com/wp-content/uploads/Apogee_Series_Leaflet.pdf. (consulté le 21/04/2022).
- [7] SBG Systems. *Ellipse Series Leaflet*. URL: https://www.sbg-systems.com/wp-content/uploads/Ellipse_Series_Leaflet.pdf. (consulté le 21/04/2022).
- [8] SBG Systems. *Ellipse Test Result*. URL: https://www.sbg-systems.com/wp-content/uploads/2018/07/Ellipse_INS-Automotive-Test-Result.pdf. (consulté le 21/04/2022).
- [9] Jean-Marie Truffaut. “Etude expérimentale de l’origine du bruit émis par les flammes de chalumeaux”. In: *Archives ouvertes.fr* (Feb. 2006), pp. 21–54. URL: <https://tel.archives-ouvertes.fr/tel-00011688/document>.

Annexe

Pour une meilleure lisibilité, l'intégralité du code écrit pour ce PSC est disponible sur GitHub à l'adresse suivante :

<https://github.com/ejilly/PSC>