Université HASSAN II de Casablanca

École Supérieure de technologie

Département Génie Électrique

Rapport de projet de fin d'études

Filière : Génie Électrique

Option : Électronique et informatique industrielle

Intitulé

Réalisation d'une canne intelligente avec sa propre application mobile pour la navigation des malvoyants

Realisé par :

EL KHAYDER Zakaria

OUCHATO Iliass

Encadré par :

Pr. DKHICHI Fayrouz

Année Universitaire

2021-2022

Made with LaTeX

Remerciement

Au terme de ce travail, on tient à exprimer notre profonde gratitude à notre chère professeur et encadrante **Pr. DKHICHI Fayrouz** pour son suivi et pour son énorme soutien tout au long de la période du projet.

On tient à remercier également **M. LAROUI Abderrahim** et **M. BOUTAKAOUA BACHIR Mohamed** pour le temps qu'ils ont consacré et pour les précieuses informations lors de la conception et la réalisation de notre projet.

On adresse aussi nos vifs remerciements aux membres des jurys pour avoir bien voulu examiner et juger ce travail.

On ne laissera pas cette occasion passer, sans remercier tous les enseignants et le personnel de **L'École supérieure de technologie de Casablanca (ESTC)**, et particulièrement ceux du département électrique pour leur aide et leurs précieux conseils et pour l'intérêt qu'ils portent à notre formation.

Enfin, nos remerciements à tous ceux qui ont contribué de prés ou de loin au bon déroulement de ce projet de fin d'étude.

Dédicace

Nous dédions ce modeste travail à :

— En premier lieu ceux que personne ne peut compenser les sacrifices qu'ils ont consentis pour notre éducation et notre bien-être à nos parents qui se sont sacrifiés pour nous prendre en charge tout au long de notre formation et qui sont l'origine de notre réussite que dieu les garde et les protèges.

— A notre famille et nos chers amis qui nous ont accordé leur soutien dans les instants les plus difficiles.

— Tous nos formateurs et toute l'équipe pédagogique et administrative de ESTC pour l'aide qu'ils ont toujours porté aux étudiants.

— Toute personne qui de près ou de loin a participé à notre formation.

Table des matières

Remerciement I

Dédicace II

Table des matières III

Liste des figures VII

Liste des tableaux X

Liste des codes sources XI

Glossaire XII

1 Présentation du projet 1

1.1 Introduction 1

1.2 Problématique et besoins 1

1.3 Les taches intelligents à réaliser 2

1.3.1 La détection d'obstacles 3

1.3.1.1 Le HC-SR04 3

1.3.1.1.1 Description 3

1.3.1.1.2 Avantages 3

1.3.1.1.3 Principe de fonctionnement 3

1.3.1.2 Vibreur 5

1.3.1.3 Motor Driver L293D 6

1.3.2 La navigation 6

1.3.2.1 Le HC-05 7

1.3.2.1.1 Description 7

1.3.2.1.2 Brochage 7

1.3.3 Localisation de la place en cas de perte 7

1.3.4 Trouver la canne en utilisant le téléphone et vice-versa 8
1.3.4.1 Bipeur (Buzzer) 8
1.4 Arduino 9
1.4.1 C'est Quoi un Arduino ? 9
1.4.2 Les différents types d'un Arduino ? 9
1.4.3 Que-ce-qu'on a choisi ? 11
1.5 Conclusion 12

2 Conception et expérimentation **13**
2.1 Les technologie utilisés 13
2.1.1 Les Langues de programmation 13
2.1.2 Les applications et outils 14
2.1.3 Les services utilisés 14
2.1.3.1 GeoCoding 15
2.1.3.2 Nearby Search 17
2.1.3.3 Place Details 19
2.2 La communication Bluetooth 22
2.2.1 Établissions de la connexion 22
2.2.1.1 Au niveau du téléphone 22
2.2.1.2 Au niveau de la canne 23
2.2.2 Lecture Traitement des instructions 24
2.2.2.1 Lecture de l'emport 25
2.2.2.1.1 Au niveau du téléphone 27
2.2.2.1.2 Au niveau de la canne 27
2.2.2.2 Traitement de l'instruction 28
2.2.2.2.1 Au niveau du téléphone 28
2.2.2.2.2 Au niveau de la canne 29
2.2.3 L'envoie d'une instruction 30
2.2.3.1 Au niveau de la canne 30
2.2.3.2 Au niveau de l'application mobile 31
2.3 Feedback (rétroaction) 31
2.3.1 Visual feedback 31
2.3.2 Audible feedback 32
2.3.3 Vocal feedback 34

2.4 Les boutons de commande 35
2.5 Les taches intelligents réalisés sur la canne 38
2.5.1 La détection d'obstacles 38
2.5.1.1 Montage 38
2.5.1.2 Fonctionnement 40
2.5.1.3 Des essais 40
2.5.1.4 La vibration 42
2.5.1.5 Le problème avec l'approche actuelle 44
2.5.1.6 Conclusion 50
2.5.2 La navigation 51
2.5.2.1 Les endroits à proximité 51
2.5.2.2 Recherche d'un endroit voulu 54
2.5.2.3 Les endroits favoris 57
2.5.2.4 Partage des endroits 57
2.5.2.5 La navigation 59
2.5.3 Localisation de la place en cas de perte 59
2.5.4 Trouver le téléphone avec la canne et vice-versa 61
2.6 Le site web 62
2.7 Conception de la canne 68
2.7.1 Introduction 68
2.7.2 Conception des circuits imprimés 68
2.7.2.1 Les circuits imprimés 68
2.7.2.2 Design 68
2.7.2.3 Circuit arduino 71
2.7.2.4 Circuit des boutons 79
2.7.3 Conception 3D de bras 82
2.7.3.1 Impression 3D 84

3 Finalisation et réalisation de la canne 125

3.1 Les étapes de la réalisation d'un circuit imprimé 125
3.1.1 Découpe 125
3.1.2 Insolation 126
3.1.3 Immersion 126
3.1.4 Gravure 127

3.1.5 Nettoyage 127
3.1.6 Perçage 127
3.2 PCB du circuit des boutons 128
3.3 PCB du circuit Arduino 130
3.4 L'impression 3D 130
3.5 Le coût du projet 132

Webographie **133**

Table des figures

1.1 Principe du fonctionnement du HC-SR04 4
1.2 Les signales de commande et de réponse du HC-SR04 4
1.3 Brochage du module SR-HC04 5
1.4 Moteur Vibreur 5
1.5 Brochage du driver moteur L293D [5] 6
1.6 Module Bluetooth HC-05 7
1.7 Cycle simplifié d'envoi du SMS au cas de perte 8
1.8 Buzzer 9
1.9 Arduino Nano 12

2.1 Les différents langues de programmation utilisées 13
2.2 Logigramme simplifié de la communication Bluetooth 22
2.3 Schéma de montage de HC-05 avec l'arduino 24
2.4 Logigramme de lecture du Payload Bluetooth 26
2.5 Schéma de montage des LEDs avec l'Arduino 32
2.6 Schéma de montage de bipeur avec l'Arduino 33
2.7 Schéma et montage de la connexion entre HC-SR04 avec l'arduino 39
2.8 Logigramme du code de HC-SR04 40
2.9 Essais pratique de HC-SR04 41
2.10 Vitesse du son en fonction de la température 42
2.11 Exemple d'un signal PWM 43
2.12 Sortie PWM en fonction de la proximité d'obstacle 43
2.13 Essais de la sortie PWM en fonction de la distance 44
2.14 Schéma avec la vibration de la connexion entre HC-SR04 avec l'arduino 44
2.15 Logigramme de notre principe personnalisé de fonctionnement de HC-SR04 45
2.16 Diagramme de notre principe personnalisé de fonctionnement de HC-SR04 avec le chien de garde 46
2.17 Diagramme simplifié de la recherche des endroits à proximité 51

2.18 Recherche des endroits à proximité 53
2.19 Diagramme simplifié de la recherche d'un endroit voulu 54
2.20 Recherche d'un endroit voulu 56
2.21 Les endroits favoris 57
2.22 Partage des endroits 58
2.23 Logigramme de l'SMS en cas de perte 59
2.24 SMS de position actuelle en cas de perte 61
2.25 Website : Version Web 66
2.26 Website : Version mobile 67
2.27 Un exemple d'un PCB 68
2.28 Interface du logiciel Altium Designer 69
2.29 Problème de plantage de Altium Designer 70
2.30 Interface Proteus 70
2.31 Circuit d'Arduino 71
2.32 Circuit Proteus d'Arduino 72
2.34 Le package d'Arduino | Le support femelle 73
2.35 Paramètres de support défaut de Proteus et les spécifications réelles 74
2.36 Layout du PCB du circuit Arduino 74
2.37 Modèle 3D du PCB du circuit Arduino 75
2.38 Circuit d'Arduino finale 75
2.39 Le L293D 76
2.40 Le PCB du circuit Arduino avec le L293D 76
2.41 Modèle 3D PCB du circuit Arduino avec le L293D 76
2.42 Le circuit final Arduino en Proteus 78
2.43 Layout du PCB final Arduino en Proteus 78
2.44 Modèle 3D du PCB final d'Arduino 79
2.54 Sketch 2D de la première itération de la canne 85
2.57 Première version du modèle 3d du circuit imprimé des boutons dans SolidWorks 91
2.58 Première version du modèle 3D de la bras 92
2.61 Circuit boutons 3d avec les bouchons 93
2.62 Première itération du bras 94

3.1 Châssis d'insolation 126
3.2 Graveuse CIF bb4 127

3.3 Perceuse 128
3.4 L'immersion du circuit imprimé des boutons 128
3.5 Perçage du circuit imprimé des boutons 129
3.6 Boutons soudés circuit imprimé des boutons 129
3.7 Circuit imprimé final des boutons 130
3.8 La réalisation du circuit d'Arduino 130
3.9 Fiches de la fabrication additive 132

Liste des tableaux

1.1 Les différents types d'Arduino [8] 11

2.1 Prix des services Google Places [9] 14
2.2 Liste des instructions Bluetooth 25
2.3 Cartographie des boutons 38

3.1 Les charges pour la réalisation du projet 132

Liste des Codes Sources

2.1 Exemple d'une réponse Google GeoCoding 16
2.2 Exemple d'une réponse Nearby Search 18
2.3 Exemple d'une réponse Place Details 22
2.4 Connecter le téléphone à HC-05 23
2.5 Connecter le canne à HC-05 23
2.6 Lecture du Payload Bluetooth au niveau de l'application mobile 27
2.7 Lecture du Payload Bluetooth au niveau de l'Arduino 27
2.8 Classe gestionnaire d'instruction parent de l'application mobile 28
2.9 Traiteur du Payload Bluetooth de l'application mobile 29
2.10 Classe gestionnaire d'instruction parent de la canne 29
2.11 Traiteur du Payload Bluetooth de la canne 30
2.12 Envoyer une instruction d'après la canne 31
2.13 Envoyer une instruction au niveau de l'application mobile 31
2.14 Ringtone 34
2.15 Button 38
2.16 HC-SR04 49
2.17 WatchDog 49
2.18 Vibreur 50
2.19 Search for nearby places 52
2.20 Recherche d'un endroit voulu 55
2.21 Expéditeur d'SMS d'urgence 61
2.22 Website 65

Glossaire

API Application Programming Interface. 14

ASCII American Standard Code for Information Interchange. 25

DAC Digital/Analog Converter. 9

ESTC L'École supérieure de technologie de Casablanca. I, II

GPS Global Positioning System. 59

HTTP Hypertext Transfer Protocol. 14

IDE Integrated Development Environment. 14

JSON JavaScript Object Notation. 14

LED Light-emitting diode. VII, 31, 32

OOP Object Oriented Programming. 33

PCB Printed Circuit Board. 14, 68

PWM Pulse Width Modulation. VII, 10, 32, 42–44

RAM Random Access Memory. 9

SDK Software Development Kit. 13

WHO L'organisation Mondiale de la Santé. 1, 133

1 | Présentation du projet

1.1 Introduction

La nature a donné à l'homme et aux animaux complexes des organes qui leur permettent d'interpréter les différentes informations de leur environnement. Ces organes peuvent capter des grandeurs physiques qui sont envoyées au cerveau pour pouvoir interpréter les changements dans le monde qui nous entoure ; ce sont les sens. L'un des phénomènes physiques capté par nos organes est "les ondes", tantôt mécaniques avec l'ouïe, tantôt électromagnétiques avec la vue, qui constituent les deux principaux sens de l'homme. Or ces sens ne captent qu'une infime partie de tout le spectre existant des ondes.

Malheureusement ces sens peuvent être endommagés, ce qui devient une contrainte pour la personne affectée. Selon L'organisation Mondiale de la Santé (WHO), environ 1,3 milliard de personnes vivrait avec une forme de déficience visuelle [1], où, approximativement, 49.1 million d'eux sont aveugles [2].

Les personnes aveugles, ont beaucoup de difficultés non seulement pour se déplacer physiquement, en particulier dans des environnements qu'ils ne connaissent pas, et encore devant des obstacles qu'ils ne voient pas.

1.2 Problématique et besoins

Heureusement, il existe différentes techniques, outils et des technologies disponibles pour permettre aux handicapés de réaliser leurs activités quotidiennes, Un des outils les plus utilisés est la canne blanche qui permet à l'utilisateur de détecter des obstacles qui se trouvent à un mètre de lui environ et, également, de déceler l'état du sol sur lequel ils marchent. Cependant, il existe encore des déficiences pour détecter des obstacles plus hauts ainsi que pour toucher les obstacles et les reconnaître. La technologie des capteurs nous permet d'identifier des grandeurs physiques et de les

transformer en informations grâce à la connaissance des phénomènes physiques qui y interviennent. La compréhension des ondes a permis de développer des capteurs qui peuvent détecter des objets à distance grâce à l'analyse des échos reçus.

Mais il reste encore le problème de déplacement, Les personnes aveugles sont les plus susceptibles de se perdre, car elles ne peuvent pas localiser les lieux en raison de leur manque de vue c'est pourquoi on a pensé à élargir la portée de la canne blanche, l'utilisation de la technologie nous a poussé à introduire le GPS (Global Positioning System) , au premier lieu cette idée va sembler compliquée mais on a pensé à utiliser un système de navigation nous l'avons tous chez nous c'est le smartphone , Tous les smartphones commercialisés ces dernières années sont dotés d'une puce GPS et c'est grâce à celle-ci que votre téléphone peut être géolocalisé, c'est-à-dire que vous pouvez connaître sa position géographique, alors on va essayer de créer une liaison entre eux et la canne blanche pour qu'il nous aide à résoudre le problème de navigation chez les aveugles
et pour rendre notre système plus parfait on va essayer de traiter d'autres problèmes qui rencontre les aveugles comme :

- Perdre leurs cannes qui va être un problème sérieux car elle est la seule méthode de leurs navigations et cela le rend vulnérables aux accidents Ce qui les rend vulnérables aux accidents, et pire encore, les rendra devant le problème que nous avons mentionné plus tôt, qui est de se perdre dans un endroit qu'ils ne connaissent pas, et encore une fois on va utiliser le smartphone pour trouver la canne et vice versa.
- la décharge d'un des composants (canne blanche, smartphone).
- l'utilisation de smartphone, on ne doit pas oublier la faite que les aveugles vont trouver des difficultés a utiliser leurs smartphones notre rôle va être crée une méthode de communication entre les aveugles et leurs smartphones basée sur l'ouïe et le toucher.

1.3 Les taches intelligents à réaliser

Comme indiqué précédemment, les aveugles ont du mal à faire leurs tâches quotidiennes simples, sans parler de pouvoir d'aller dehors. Ce qui nous a fait réfléchir à la façon dont nous pouvons faciliter leur navigation. Alors que les objectifs de notre projet sont les suivants :

- La détection des obstacles et leur proximité.
- Faciliter l'utilisation du téléphone pour la navigation et la rendre plus rapide et plus facile.
- Être en mesure de trouver leur canne égarée à l'aide de leur téléphone, et vice versa.
- Appeler à l'aide au cas où il serait perdu, et généralement faciliter l'information des autres

de leur emplacement actuel.

— Trouver les différents lieux proches de lui, classés par catégories, et l'y conduire.
— Contrôler toutes ces tâches avec son téléphone et/ou sa canne.
— Avoir la possibilité de charger son téléphone avec la canne

1.3.1 La détection d'obstacles

Pour réaliser cette tache, on utilisera un capteur de proximité qui permet la détection d'obstacles et leur proximité. À cause de son prix, sa portabilité, ainsi que sa précision et sa simplicité d'utilisation, on a choisi le modèle HC-SR04 à utiliser.

1.3.1.1 Le HC-SR04

1.3.1.1.1 Description Le capteur de distance ultrasonique HC-SR04 est un capteur utilisé pour détecter la distance d'un objet à l'aide d'un sonar. Il est idéal pour tous les projets de robotique que vous avez qui vous obligent à éviter les objets, en détectant à quel point ils sont proches, vous pouvez vous éloigner d'eux [3] !

1.3.1.1.2 Avantages Le capteur est petit, facile à utiliser dans n'importe quel projet de robotique et offre une excellente détection sans contact entre 2 cm à 400 cm avec une précision de 3mm. Puisqu'il fonctionne sur 5 volts, il peut être raccordé directement à un Arduino ou tout autre microcontrôleur logique 5V.

1.3.1.1.3 Principe de fonctionnement Tout commence lorsqu'une impulsion d'au moins 10 µs (10 microsecondes) de durée est appliquée à la broche de déclenchement (Trigger). En réponse à cela, le capteur transmet une rafale sonore de huit impulsions à 40 KHz. Ce motif à 8 impulsions rend la « signature ultrasonique » de l'appareil unique, ce qui permet au récepteur de différencier le motif transmis du bruit ultrasonique ambiant.

Les huit impulsions ultrasoniques se déplacent dans l'air loin du transmetteur. Pendant ce temps, la broche Echo va HAUT pour commencer à former le début du signal d'écho-retour.

Dans le cas, si ces impulsions ne sont pas réfléchies en arrière, le signal Echo expire après 38 ms (38 millisecondes) et revient bas. Ainsi, une impulsion de 38 ms n'indique aucune obstruction dans la plage du capteur [4].

On peut calculer la distance entre le capteur et l'objet en utilisant la simple relation en fonction de

la vitesse du sons et la durée de déclenchement :

$$d = \frac{v \cdot t}{2} \text{ où } v \approx 340m/s$$

En devise par 2 car le temps t est le temps d'aller-retour.

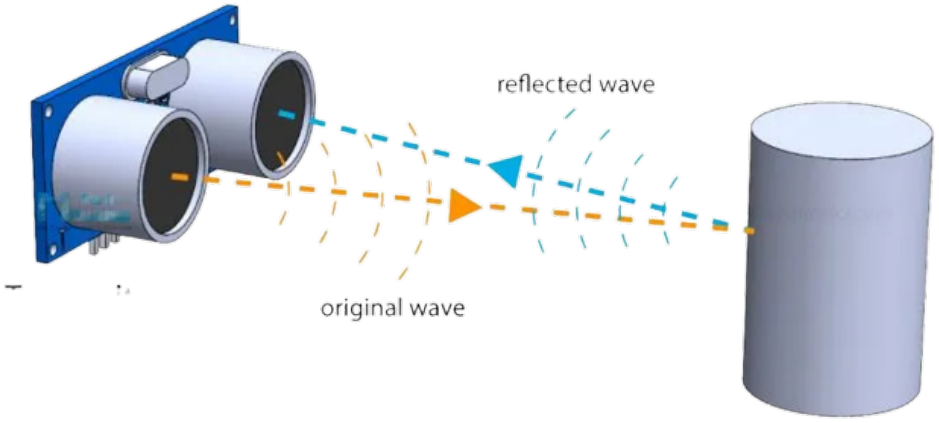


FIGURE 1.1 – Principe du fonctionnement du HC-SR04

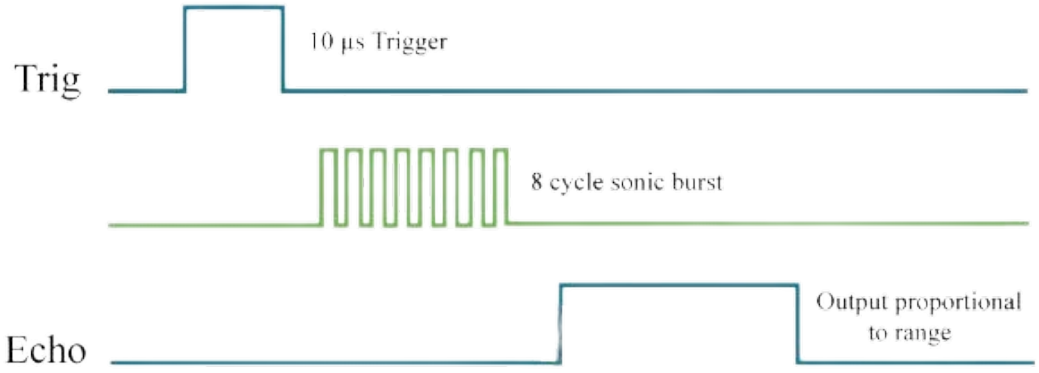


FIGURE 1.2 – Les signales de commande et de réponse du HC-SR04

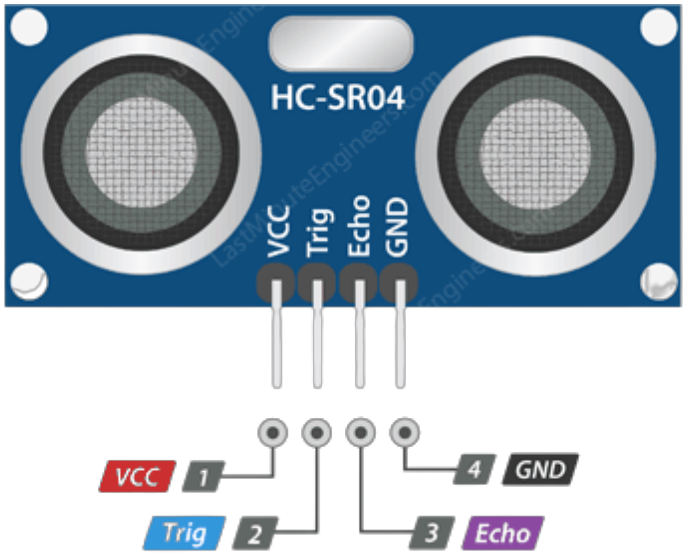


FIGURE 1.3 – Brochage du module SR-HC04

1.3.1.2 Vibreur

Pour informer l'utilisateur qu'il y a un obstacle devant lui, on utilisera deux vibreurs, cités à chaque côté de la canne. L'intensité de vibration augmente avec la proximité d'obstacle.

FIGURE 1.4 – Moteur Vibreur

Ce type de vibreurs nécessite 60mA et 5V pour bien fonctionner, main l'Arduino nano a une limitation de 40mA par broche, d'où on a besoin d'un motor driver.

1.3.1.3 Motor Driver L293D

Le circuit intégré L293D permet de piloter 2 moteurs à courant courant continu dans les deux sens de rotation et pour faire de la variation de vitesse.

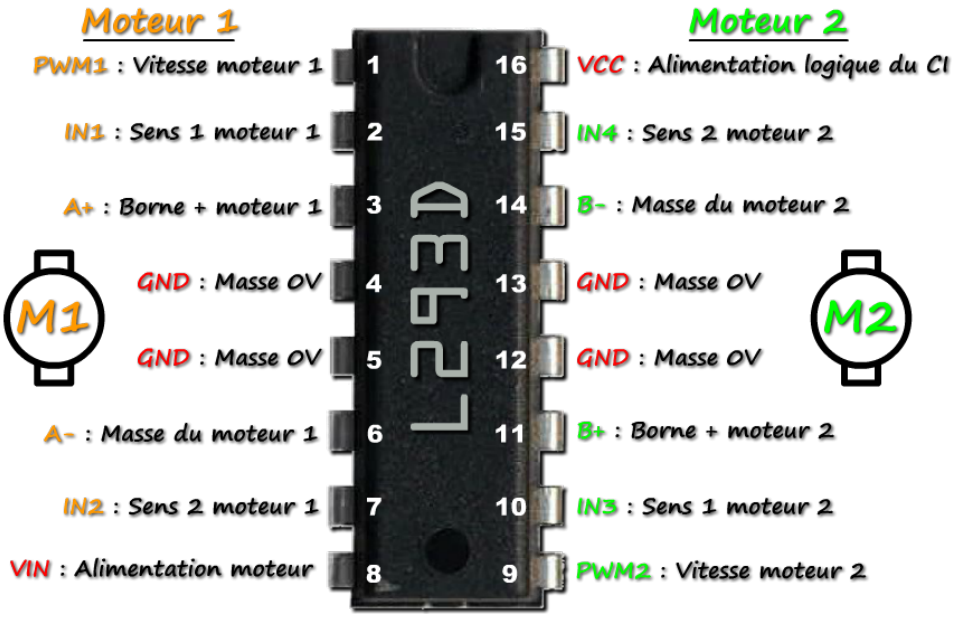


FIGURE 1.5 – Brochage du driver moteur L293D [5]

Pour notre projet, on n'a pas besoin de tourner le moteur à deux sens, donc on peut utiliser jusqu'à 4 moteurs.

1.3.2 La navigation

Pour aider l'utilisateur lors de la navigation, nous allons créer une application mobile multi-plateforme. L'application aidera l'utilisateur à trouver des endroits qu'il connaît déjà, ou à découvrir des endroits à proximité de différentes catégories : Restaurants, mosquées, magasins et services, même les médecins et les pharmacies. Et si jamais il a aimé un endroit, il peut l'ajouter comme une place favourite pour qu'il puisse le retrouver simplement la prochaine fois.

L'application sera également responsable de guider l'utilisateur à sa destination, en utilisant Google Places API, et l'application Google Map.

Il sera également utilisable mains libres, juste par les boutons sur la canne.

La communication entre la canne et l'application mobile sera conduite par Bluetooth, en utilisant le module HC-05.

1.3.2.1 Le HC-05

1.3.2.1.1 Description Le module HC-05 est un module Bluetooth SPP (Serial Port Protocol), ce qui signifie qu'il communique avec l'Arduino via la communication série

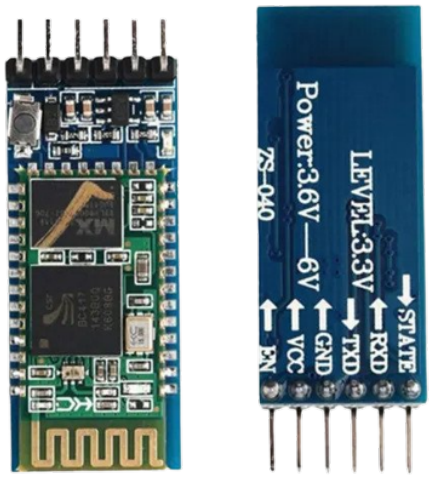


FIGURE 1.6 – Module Bluetooth HC-05

1.3.2.1.2 Brochage

Le HC-05 consiste 6 broches :

- VCC et GND : Pour l'alimentation +5V
- TXD et RXD : Pour la communication série
- State : Signifie si le module et connecter en Bluetooth ou pas. HIGH si oui, LOW si non.
- Enable : Broche de configuration, on la met a l'état HIGH pour avoir la possibilité de configurer le module (Nom, Mot de passe, Type ...) avec la communication série.

1.3.3 Localisation de la place en cas de perte

À l'aide de notre application, l'utilisateur aura la possibilité d'informer ses roches de sa position actuelle en utilisant le service GPS intégré à son smart-phone.

Sa localisation actuelle sera envoyée par un SMS aux contacts qu'il a déjà marqués comme contacts d'urgence, aussi que le message envoyer peut-être modifier simplement à l'aide de l'interface mobile.

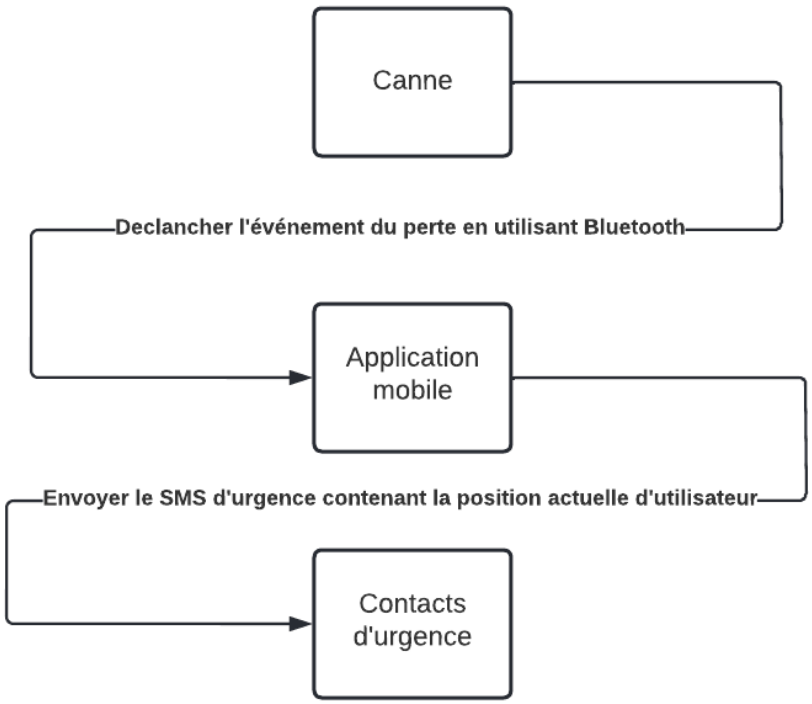


FIGURE 1.7 – Cycle simplifié d'envoi du SMS au cas de perte

1.3.4 Trouver la canne en utilisant le téléphone et vice-versa

Les personnes visuellement impayées ont du mal à trouver leurs objets, en particulier leurs téléphones. Nous allons essayer d'éliminer ce problème en ajoutant la possibilité de commencer sonnerie de téléphone à partir de sa canne. Et ils peuvent également trouver leurs cannes en utilisant leurs téléphones en commençant une sonnerie sur la canne.

La sonnerie de la canne va être joue par le bipeur (Buzzer)

1.3.4.1 Bipeur (Buzzer)

Un buzzer (anglicisme) ou bipeur est un élément électromécanique ou piézoélectrique qui produit un son caractéristique quand on lui applique une tension : le bip. Certains nécessitent une tension continue, d'autres nécessitent une tension alternative [6].

FIGURE 1.8 – Buzzer

1.4 Arduino

1.4.1 C'est Quoi un Arduino ?

Arduino est une plateforme électronique open-source basée sur du matériel et des logiciels faciles à utiliser. Cartes Arduino sont capables de lire les entrées - lumière sur un capteur, un doigt sur un bouton, ou un message Twitter - et de le transformer en une sortie - activer un moteur, allumer une LED, publier quelque chose en ligne. Vous pouvez dire à votre carte quoi faire en envoyant un ensemble d'instructions au micro-contrôleur de la carte [7].

1.4.2 Les différents types d'un Arduino ?

Les Arduinos sont caractérisés par plusieurs caractéristiques :

- Processeur
- Vitesse d'horloge
- Tension d'entrée
- Tension de sortie
- Mémoire de stockage (Flash Memory)
- Mémoire vive (RAM)
- Nombre des pins entrée/sortie logique
- Nombre des pins entrée/sortie analogique
- Nombre des convertisseurs numérique/analogique (DAC)

— Nombre des pins supportant la modulation de largeur d'impulsion (PWM)

— Les différents protocoles de communication supportés (UART / SPI / I^2C)

— Type du connecteur

— Ses dimensions

Pour notre projet, on n'est pas très intéressé à toutes ses différentes caractéristiques, d'où on a créé un tableau qui compare les diffèrent types en se basant sur les caractéristiques qu'on a besoin.

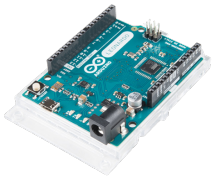
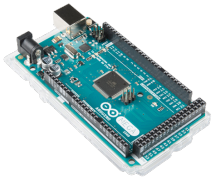
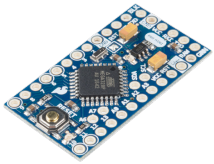
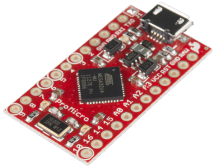
	Modèle	Vitesse d'horloge	Mémoire de programme	RAM	Nombre des pins	PWM
	UNO	16MHz	32KB	2KB	20	6
	Nano	16MHz	32KB	2KB	20	6
	Mega	16MHz	256KB	8KB	54	15
	Pro Mini	8MHz	32KB	2KB	22	8
	Pro Micro	16MHz	32KB	2.5KB	18	5
	Leonardo	16MHz	32KB	2.5KB	20	12

TABLE 1.1 – Les différents types d'Arduino [8]

1.4.3 Que-ce-qu'on a choisi ?

Notre projet a besoin de 14 entrées/sorties logiques, et une analogique. Où on avait le choix entre plusieurs modèles comme l'Arduino UNO, Nano, Pro Mini, et toute autre diffèrent type. Donc on

avait pris autres caractéristiques en considération, surtout le prix et les dimensions, c'était indispensable d'avoir une qui peut être contenue dans un petit boîtier avec un bon prix. D'où on a choisi l'**Arduino Nano**, qu'il y avait des dimensions de 45 x 18 x 18 mm et un poids de 7g, à seulement 60DH.

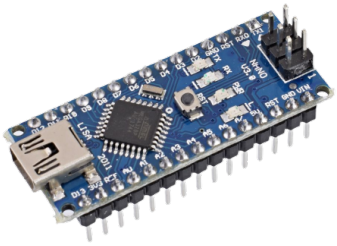


FIGURE 1.9 – Arduino Nano

1.5 Conclusion

Maintenant et après qu'on a traité se système de différente partie et étudier un peu près tous les problèmes qu'il peut rencontrer, on peuvent réaliser ce projet commençant par les taches intelligents (La détection d'obstacles, La navigation, Localisation de la place en cas de perte, Trouver la canne en utilisant le téléphone et vice-versa), la réalisation du modèle de la canne, l'assemblage.

2 | Conception et expérimentation

2.1 Les technologie utilisés

Afin de realiser notre project, on a besoin d'une arsenal d'outils.

2.1.1 Les Langues de programmation

On va utiliser certaines langues de programmations, chacune est visée vers une plate-forme spécifique :

- `Dart` avec le SDK `Flutter` : Utilisé pour la programmation de l'application mobile, on va parfois utiliser autres langages pour modifier quelques caractéristiques natives de l'application mobile qu'on peut pas modifier avec Dart seulement.
  - `Java` pour l'Android
  - `Swift` pour l'IOS
- `Html`, CSS et `JavaScript` : Pour le site web qui sera partagé par l'utilisateur, on va aussi utiliser la librairie `React.js` avec `TypeScript`.
- `Arduino` C++ : Pour le développement de l'application de la carte Arduino. Ce dernier semble à C++ standard, avec quelques différences mineures.

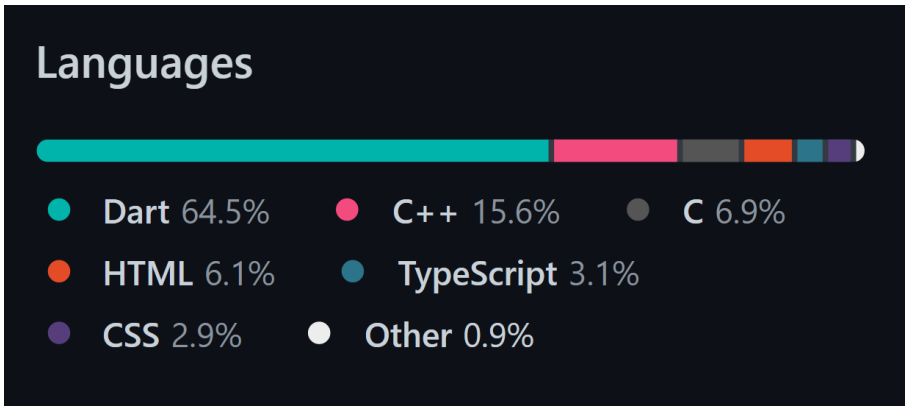


FIGURE 2.1 – Les différents langues de programmation utilisées

Service	Prix par 1000 requêtes
GeoCoding	$5
Nearby Search	$17
Place Details	$17

TABLE 2.1 – Prix des services Google Places [9]

2.1.2 Les applications et outils

Juste comme les langues, on a besoin de quelques applications et outils afin de réaliser ce projet.

— `Visual Studio Code` : C'est l'éditeur de code de choix. Il est facile, utile, et robuste.
— `Arduino` IDE : Utiliser pour compiler et transférer le code C++ vers l'Arduino.
— `SolidWorks` : Utiliser pour la conception du boîtier de la canne.
— `Fritzing` : Pour la modélisation du circuit électrique.
— `Proteus PCB` : Pour la réalisation du circuit de la carte imprimée PCB.
— `Desmos` : Pour dessiner les courbes citées dans ce rapport.

2.1.3 Les services utilisés

Pour pouvoir chercher des endroits et obtenir leur nom, leur emplacement, leur type, leurs heures d'ouverture..., nous aurons besoin d'une sorte de base de données contenant toutes ces informations, et c'est là que le service Google entre en jeu.

Google offre un vaste éventail de services, mais nous utiliserons trois d'entre eux :

— GeoCoding
— Find Place
— Place Details

Malheureusement, aucun de ces services n'est gratuit. Nous devrons donc créer une clé API pour les utiliser. mais autant qu'un étudiant, Google ne vous facture pas si l'utilisation totale est inférieure à 200 dollars.

Toute communication avec les API de Google sera fait par HTTP avec une réponse de format JSON, mais pour simplicité, parfois on va utiliser des libraires qui facilitent la communication.

2.1.3.1 GeoCoding

GeoCoding (géocodage) est le processus de conversion des adresses (comme « 1600 Amphitheatre Parkway, Mountain View, CA ») en coordonnées géographiques (comme 37.423021 de latitude et -122.083739 de longitude) que vous pouvez utiliser pour placer des marqueurs sur une carte ou positionner la carte.

Pour notre application, on utilisera souvent l'inverse (Reverse GeoCoding) pour convertir des coordonnées géographiques en une adresse lisible par l'humain [10].

Exemple d'une réponse de `Google GeoCoding` des cordonnées :

— Latitude : 33.591022

— Longitude : 3.591022


```
{
   "address_components": [
      {
         "long_name": "HHR4+CX",
         "short_name": "HHR4+CX",
         "types": ["plus_code"]
      },
      {
         "long_name": "Casablanca",
         "short_name": "Casablanca",
         "types": ["locality", "political"]
      },
      {
         "long_name": "Casablanca-Settat",
         "short_name": "Casablanca-Settat",
         "types": ["administrative_area_level_1", "political"]
      },
      {
         "long_name": "Morocco",
         "short_name": "MA",
         "types": ["country", "political"]
      }
   ],
```

```
    "formatted_address": "HHR4+CX Casablanca, Morocco",
    "geometry": {
       "bounds": {
          "northeast": {
             "lat": 33.591125,
             "lng": -7.4425
          },
          "southwest": {
             "lat": 33.591,
             "lng": -7.442625
          }
       },
       "location": {
          "lat": 33.591022,
          "lng": -7.442556000000001
       },
       "location_type": "GEOMETRIC_CENTER",
       "viewport": {
          "northeast": {
             "lat": 33.5924114802915,
             "lng": -7.441213519708498
          },
          "southwest": {
             "lat": 33.5897135197085,
             "lng": -7.443911480291503
          }
       }
    },
    "place_id": "GhIJtpvgm6bLQEARJZhqZi3FHcA",
    "plus_code": {
       "compound_code": "HHR4+CX Casablanca, Morocco",
       "global_code": "8C5JHHR4+CX"
    },
    "types": ["plus_code"]
}
```

Code 2.1 – Exemple d'une réponse Google GeoCoding

2.1.3.2 Nearby Search

La recherche à proximité vous permet de rechercher des endroits dans une zone spécifiée. Vous pouvez affiner votre demande de recherche en fournissant des mots clés ou en spécifiant le type de lieu que vous recherchez [11].

On utilisera cette service pour trouve les endroits à proximité, filtrés par type.

Exemple d'une réponse de `Google Nearby Search` :


```
{
    "html_attributions": [ ],
    "next_page_token": "Aap_uECVBJCY8nZXQb9ssfnK_LfkwdVQm...",
    "results": [
        {
            "business_status": "OPERATIONAL",
            "geometry": {
                "location": {
                    "lat": -33.8587323,
                    "lng": 151.2100055
                },
                "viewport": {
                    "northeast": {
                        "lat": -33.85739817010727,
                        "lng": 151.2112278798927
                    },
                    "southwest": {
                        "lat": -33.86009782989272,
                        "lng": 151.2085282201073
                    }
                }
            },
            "icon": "https://maps.gstatic.com/mapfiles/place...",
            "icon_background_color": "#FF9E67",
            "icon_mask_base_uri": "https://maps.gstatic.com/...",
```

```
            "name": "Cruise Bar",
            "opening_hours": {
                "open_now": false
            },
            "photos": [
                {
                    "height": 575,
                    "html_attributions": [
                        "<a href=\"https://maps.google.com/m..."
                    ],
                    "photo_reference": "Aap_uEDnz_WTtjeSoT06...",
                    "width": 766
                }
            ],
            "place_id": "ChIJi6C1MxquEmsR9-c-3O48ykI",
            "plus_code": {
                "compound_code": "46R6+G2 The Rocks, New Sout...",
                "global_code": "4RRH46R6+G2"
            },
            "price_level": 2,
            "rating": 4.1,
            "reference": "ChIJi6C1MxquEmsR9-c-3O48ykI",
            "scope": "GOOGLE",
            "types": [
                ...
            ],
            "user_ratings_total": 1184,
            "vicinity": "Level 1, 2 and 3, Overseas Passenger..."
        },
        ...
    ],
    "status": "OK"
}
```


Code 2.2 – Exemple d'une réponse Nearby Search

La réponse contient une liste contenant des informations à propos endroits à proximité.
Chaque endroit a un champ nommé `place_id`, c'est l'un qu'on va utiliser pour demander plus d'information a propos ce dernier.

2.1.3.3 Place Details

Une fois que vous avez un `place_id` à partir d'une recherche de place, vous pouvez demander plus de détails sur un établissement particulier ou un point d'intérêt en lançant une demande de détails de place.
Une demande de Place Details renvoie des informations plus complètes sur le lieu indiqué, telles que son adresse complète, numéro de téléphone, la note de l'utilisateur et les commentaires [12].

Exemple d'une réponse de `Google Place Details` :


```
{
   "html_attributions":[

   ],
   "result":{
      "address_components":[
         {
            "long_name":"48",
            "short_name":"48",
            "types":[
               "street_number"
            ]
         },
         ...
      ],
      "adr_address":"<span class=\"street-address\">48 Pirrama Rd</s...",
      "business_status":"OPERATIONAL",
      "formatted_address":"48 Pirrama Rd, Pyrmont NSW 2009, Australia",
      "formatted_phone_number":"(02) 9374 4000",
      "geometry":{
         "location":{
            "lat":-33.866489,
            "lng":151.1958561
```

```
            },
            "viewport":{
               "northeast":{
                  "lat":-33.8655112697085,
                  "lng":151.1971156302915
               },
               "southwest":{
                  "lat":-33.86820923029149,
                  "lng":151.1944176697085
               }
            }
         },
         "icon":"https://maps.gstatic.com/mapfiles/place_api/...",
         "icon_background_color":"#7B9EB0",
         "icon_mask_base_uri":"https://maps.gstatic.com/mapfi...",
         "international_phone_number":"+61 2 9374 4000",
         "name":"Google Workplace 6",
         "opening_hours":{
            "open_now":false,
            "periods":[
               {
                  "close":{
                     "day":1,
                     "time":"1700"
                  },
                  "open":{
                     "day":1,
                     "time":"0900"
                  }
               },
               ...
            ],
            "weekday_text":[
               "Monday: 9:00 AM - 5:00 PM",
               ...
            ]
```

```
        },
        "photos":[
           {
              "height":3024,
              "html_attributions":[
                 "<a href=\"https://maps.google.com/maps/co..."
              ],
              "photo_reference":"Aap_uECTSiFbO2Qg81vEQkbLMh...",
              "width":4032
           },
           ...
        ],
        "place_id":"ChIJN1t_tDeuEmsRUsoyG83frY4",
        "plus_code":{
           "compound_code":"45MW+C8 Pyrmont NSW, Australia",
           "global_code":"4RRH45MW+C8"
        },
        "rating":4.1,
        "reference":"ChIJN1t_tDeuEmsRUsoyG83frY4",
        "reviews":[
           {
              "author_name":"Mark Smith (Mark ZZZ Smith)",
              "author_url":"https://www.google.com/maps/con...",
              "language":"en",
              "profile_photo_url":"https://lh3.googleuserco...",
              "rating":5,
              "relative_time_description":"a year ago",
              "text":"Great place to visit, cafeteria great...",
              "time":1589072760
           },
           ...
        ],
        "types":[
           ...
        ],
        "url":"https://maps.google.com/?cid=10281119596374313554",
```

```
96          "user_ratings_total":932,
97          "utc_offset":600,
98          "vicinity":"48 Pirrama Road, Pyrmont",
99          "website":"http://google.com/"
100      },
101      "status":"OK"
102  }
```

Code 2.3 – Exemple d'une réponse Place Details

2.2 La communication Bluetooth

Avant de commencer à réaliser les taches requis, on doit tout d'abord établir une connexion Bluetooth entre la canne et le téléphone.

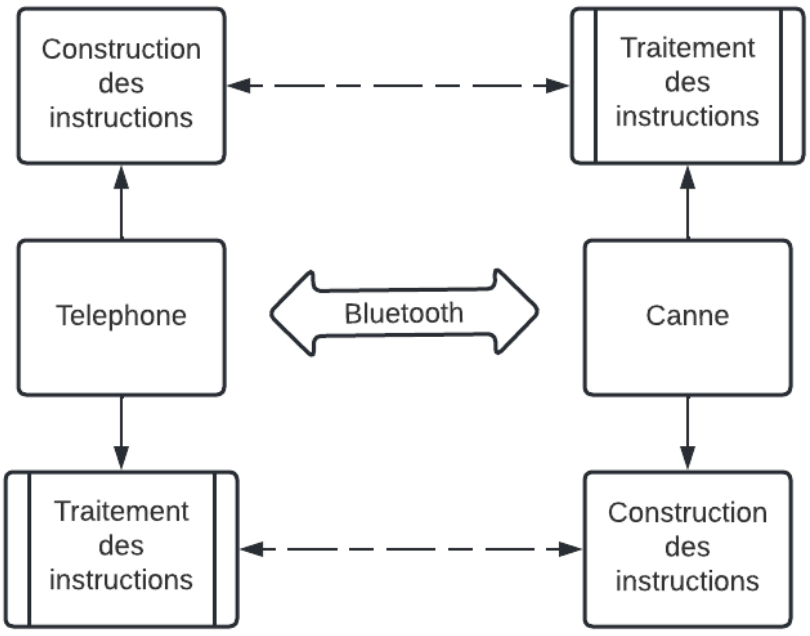


FIGURE 2.2 – Logigramme simplifié de la communication Bluetooth

2.2.1 Établissions de la connexion

2.2.1.1 Au niveau du téléphone

On doit tout d'abord établir une connexion Bluetooth au périphérique HC-05, pour simplifier un peu les chose, on utilise la librairie `flutter_bluetooth_serial` [13], puis en enregistre un auditeur

des évènements qui traite chaque instruction reçue via Bluetooth.

```
BluetoothConnection? _connection;
bool isConnected = false;
void connect() async {
    const String address = "98:D3:33:81:3D:33"; // L'adresse MAC
    try {
      _connection = await BluetoothConnection.toAddress(address);
      isConnected = true;
      _connection?.input?.listen(_onRecievePayload);
    } catch (exception) {
      isConnected = false;
    }
  }
```

Code 2.4 – Connecter le téléphone à HC-05

La fonction `_onRecievePayload` sera responsable du la lecture de l'emport Bluetooth.

2.2.1.2 Au niveau de la canne

Au niveau de la canne c'est très simple à se connecter avec le HC-05, puisque c'est une communication série, on peut utiliser la librairie SoftwareSerial.

```
#include <SoftwareSerial.h>

namespace Bluetooth
{
    SoftwareSerial BTSerial(4, 3); // TXD | RXD
}
```

Code 2.5 – Connecter le canne à HC-05

On peut utiliser les fonctions disponibles juste comme c'était l'interface `Serial`, par exemple :

- `BTSerial.print` pour écrire
- `BTSerial.read` pour lire
- ...

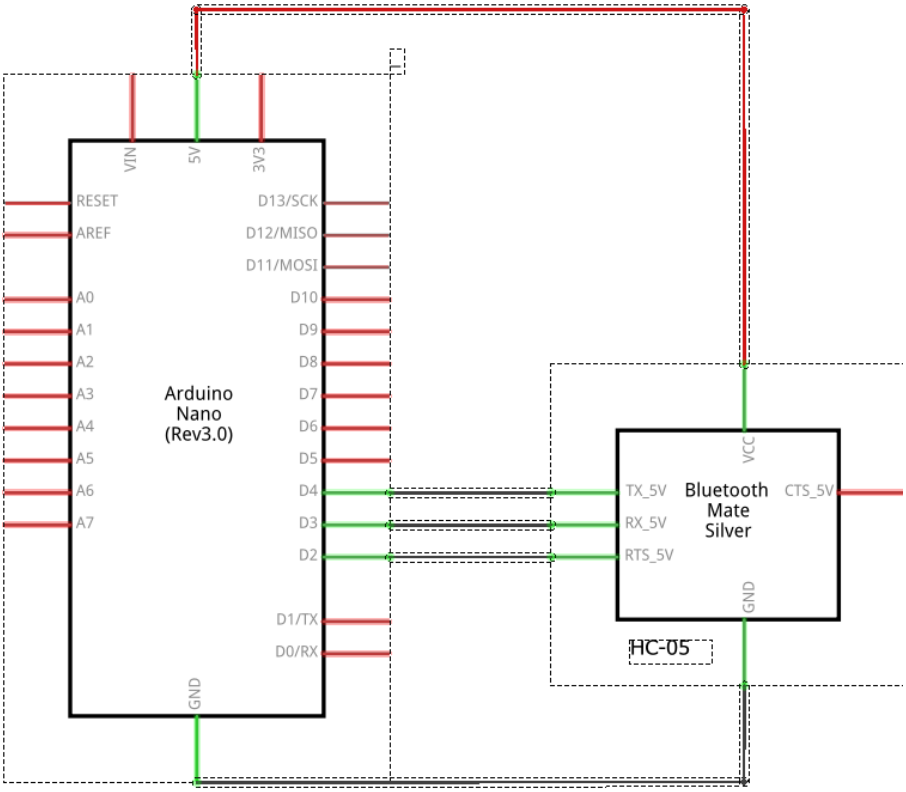


FIGURE 2.3 – Schéma de montage de HC-05 avec l'arduino

2.2.2 Lecture Traitement des instructions

Le traitement des instructions est séparé en deux parties :

- Lecture de l'emport (Payload)
- Traitement de l'instruction

Avant de commencer, on doit d'abord définir la format et une liste des instructions qui seront utilisés.

Format : `INSTRUCTION:ARG1|ARG2|ARG3|...`

Instruction	Envoyée par	Paramètres	Description
SEND_LOCATION_SMS	Canne	-	Envoyer l'SMS d'urgence
BATTERY_PERCENTAGE	Canne	pourcentage	Mettre à jour la valeur de la batterie
RING	Téléphone	-	Lancer la sonnerie de la canne
RING	Canne	-	Lancer la sonnerie du téléphone
SPEAK	Canne	phrase	synthèse vocale
NAVIGATABLES:NEXT	Canne	-	Naviguer au bouton suivant
NAVIGATABLES:PREVIOUS	Canne	-	Naviguer au bouton précédent
NAVIGATABLES:BACK	Canne	-	Retourner à la page précédente
NAVIGATABLES:PRESS	Canne	-	Cliquer le bouton sélectionné
NAVIGATABLES:RESET	Canne	-	Redémarrer l'application
NAVIGATABLES:EXPLORE	Canne	-	Naviguer à la page "Explore"

TABLE 2.2 – Liste des instructions Bluetooth

2.2.2.1 Lecture de l'emport

La communication Bluetooth est une communication série, donc on doit séparer chaque instruction de l'autre, d'où on doit fixer un caractère qui va signifier que l'instruction est terminée.
On a choisi le character \n (End of line character) qui est deja le standard, son code ASCII est 0x10.

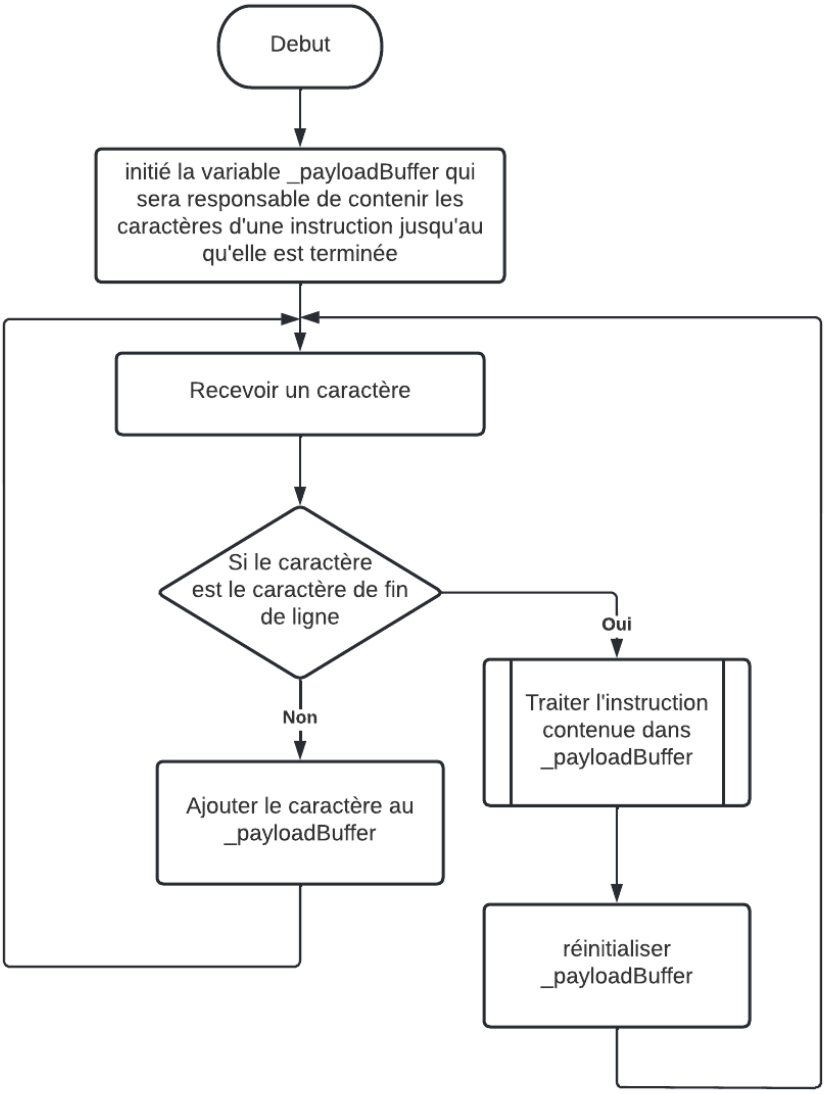


FIGURE 2.4 – Logigramme de lecture du Payload Bluetooth

2.2.2.1.1 Au niveau du téléphone

```
void _onRecievePayload(Uint8List payload) {
    for (var char in payload) {
        // 0x0A (10) corresponds to \n char
        if (char == 0x0A) {
            parseBluetoothPayload(_payloadBuffer);
            _payloadBuffer = ""; // Reset Message buffer
        } else {
            _payloadBuffer += ascii.decode([char]);
        }
    }
}
```

Code 2.6 – Lecture du Payload Bluetooth au niveau de l'application mobile

2.2.2.1.2 Au niveau de la canne

```
if (BTSerial.available()) {
    int thisChar = BTSerial.read();
    if (thisChar == 0xA) {
            parsePayload(_payloadBuffer); // Parse payload
            _payloadBuffer = "";          // Reset payload buffer
    } else {
        _payloadBuffer += ((char)thisChar); // Add current char to
        ↪ payloadBuffer after casting
    }
}
```

Code 2.7 – Lecture du Payload Bluetooth au niveau de l'Arduino

Les fonctions `parsePayload` (pour la canne) et `parseBluetoothPayload` (pour l'application mobile) seront responsable du traitement des instructions Bluetooth

2.2.2.2 Traitement de l'instruction

Pour traiter l'instruction, on doit tout d'abord la séparer pour avoir le non de l'instruction et ses arguments. Le code se diffère beaucoup entre C++ et `Dart` où il est plus facile a l'implémenter avec `Dart`.

Nous suivons un modèle de programmation orienté objet ; d'où, chaque tache aura son propre gestionnaire qui hérite les méthodes de traitement de son classe parent.

2.2.2.2.1 Au niveau du téléphone

```dart
abstract class BluetoothPayloadHandler {
  late String command;
  void handle(List<String> args);
}
```

Code 2.8 – Classe gestionnaire d'instruction parent de l'application mobile

```dart
void parseBluetoothPayload(String payload) {
  print(payload);

  final String command = payload.split(":").first;

  final List<String> args =
      payload.split(":").length == 2 ?
      ↪  payload.split(":").elementAt(1).split("|") : [];

  final List<BluetoothPayloadHandler> _handlers = [
    SendCurrentLocationSMS(),
    UpdateCaneBatteryPercentage(),
    StartPhoneRingtone(),
    Speak(),
    NavigatablesRemote(),
  ];

  for (var handler in _handlers) {
    if (handler.command == command) handler.handle(args);
```

```
19    }
20  }
```

Code 2.9 – Traiteur du Payload Bluetooth de l'application mobile

2.2.2.2.2 Au niveau de la canne

```
1   class BluetoothHandler
2   {
3   public:
4       String command;
5       void (*handle)(String[], int);
6
7       BluetoothHandler(
8           String _command,
9           void (*_handle)(String[], int))
10      {
11          command = _command;
12          handle = _handle;
13      }
14  };
```

Code 2.10 – Classe gestionnaire d'instruction parent de la canne

```
1   void parsePayload(String payload) {
2       int indexOfDoublePoints = payload.indexOf(":");
3       String command = payload.substring(0, indexOfDoublePoints);
4       String argsString = payload.substring(indexOfDoublePoints + 1);
5       String args[] = {};
6       int argsLength = 0;
7       int indexOfPipe = -1;
8       int currentPipeSearchIndex = 0;
9
10      do {
11          indexOfPipe =
            ↪  argsString.substring(currentPipeSearchIndex).indexOf("|");
```

```
12          String arg = argsString.substring(currentPipeSearchIndex,
            ↪  indexOfPipe);
13          currentPipeSearchIndex += arg.length() + 1;
14          args[argsLength] = arg;
15          argsLength++;
16      } while (indexOfPipe != -1);
17
18      for (int i = 0; i < _handlersCount; i++) {
19          BluetoothHandler handler = _handlers[i];
20          if (handler.command == command) {
21              handler.handle(args, argsLength);
22              break;
23          }
24      }
25  }
```

Code 2.11 – Traiteur du Payload Bluetooth de la canne

2.2.3 L'envoie d'une instruction

Pour envoyer une instruction, on doit la mettre en forme

2.2.3.1 Au niveau de la canne

```
1   void send(String command, String args[], int length) {
2       String payload = command + ":";
3
4       for (int i = 0; i < length; i++) {
5           payload += args[i];
6           if (i != (length - 1)) // Add pipe char (|) between args
7               payload += "|";
8       }
9
10      payload += ((char)0x0A); // Add \n char
11      BTSerial.print(payload); // Send payload
12  }
```

Code 2.12 – Envoyer une instruction d'après la canne

2.2.3.2 Au niveau de l'application mobile

```
void send(String command, List<String> args) {
    String payload = command + ":" + args.join("|");
    _sendPayload(payload);
}

void _sendPayload(String payload) {
    _connection?.output.add(
      Uint8List.fromList([...ascii.encode(payload), 0x0A]),
    );
}
```

Code 2.13 – Envoyer une instruction au niveau de l'application mobile

2.3 Feedback (rétroaction)

Comme l'utilisateur a une déficience visuelle, il n'aura aucune idée si la commande qu'il a demandée a été exécutée ou non, puisque nous devons lui fournir un certain type de rétroaction, et nous avons fait les trois.

- Visual feedback (rétroaction visuelle)
- Audible feedback (rétroaction audio)
- Vocal feedback (rétroaction vocale)

2.3.1 Visual feedback

Le visuel ne cible pas directement l'utilisateur malvoyant, c'est un signe pour les autres.
Il y aurait 3 LEDs, de couleurs différentes :

- Blanche : pour rendre l'utilisateur plus visible lors de la navigation dans la nuit, pour éviter de ne pas être vu.
- Verte : Pour indiquer que la canne est allumée
- Rouge : Pour indiquer que la batterie est déchargé (moins de 20%)

Les LEDs sont connectés à l'Arduino selon ce schéma :

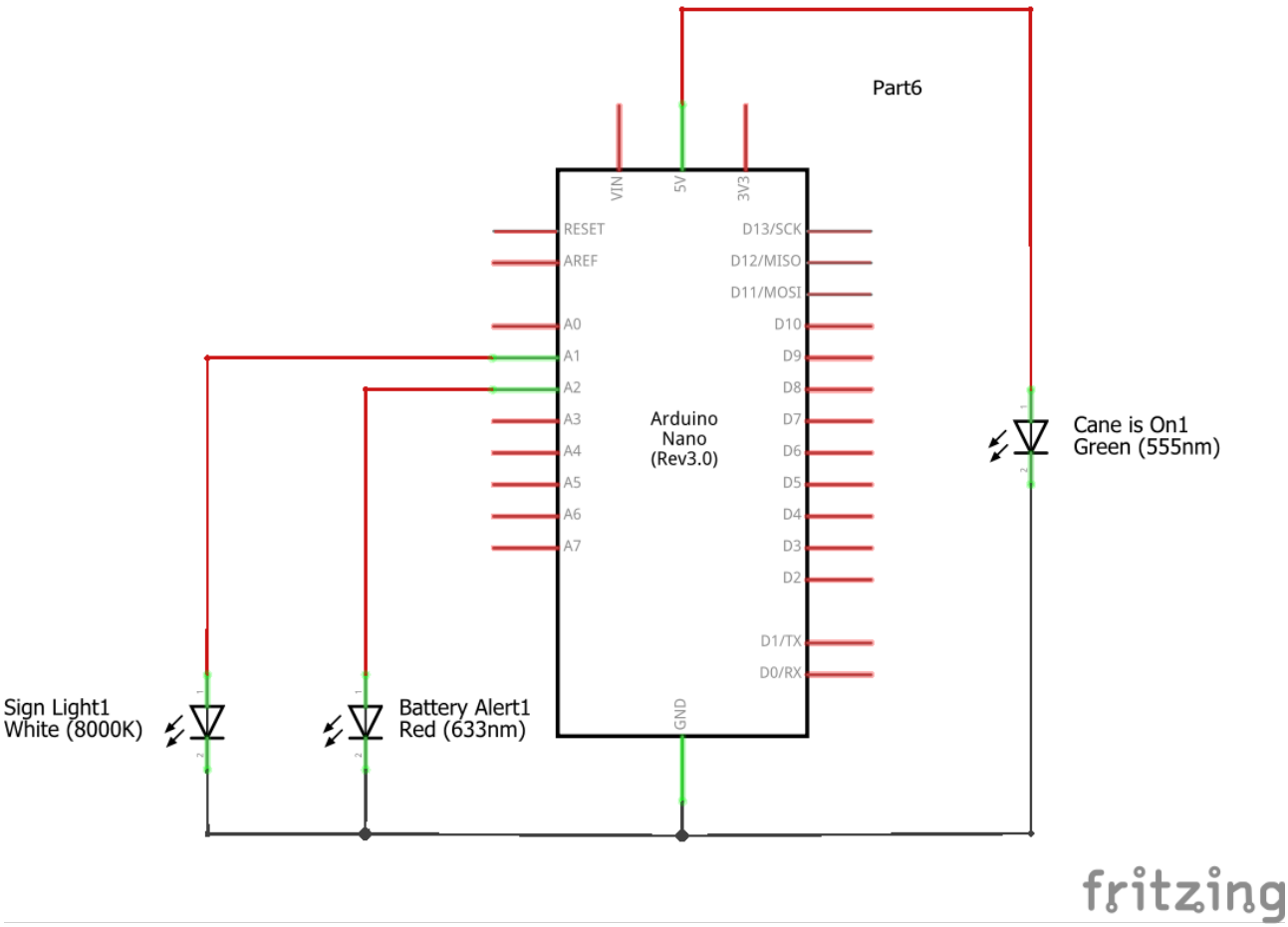


FIGURE 2.5 – Schéma de montage des LEDs avec l'Arduino

On a utilisé les broches d'entrée analogue d'Arduino (Ax) puisque les autres broches (Dx) ont manqué. Les porte A0 à A4 peut être utilisé comme sortie logique sans aucun problème

2.3.2 Audible feedback

En utilisant le Buzzer dont nous avons parlé au chapitre 1.3.4.1, nous pouvons jouer une sonnerie à l'aide de la fonction Arduino `tone` [14].

Le bipeur doit être connecter à une broche de sortie PWM, on a choisi la broche 11.

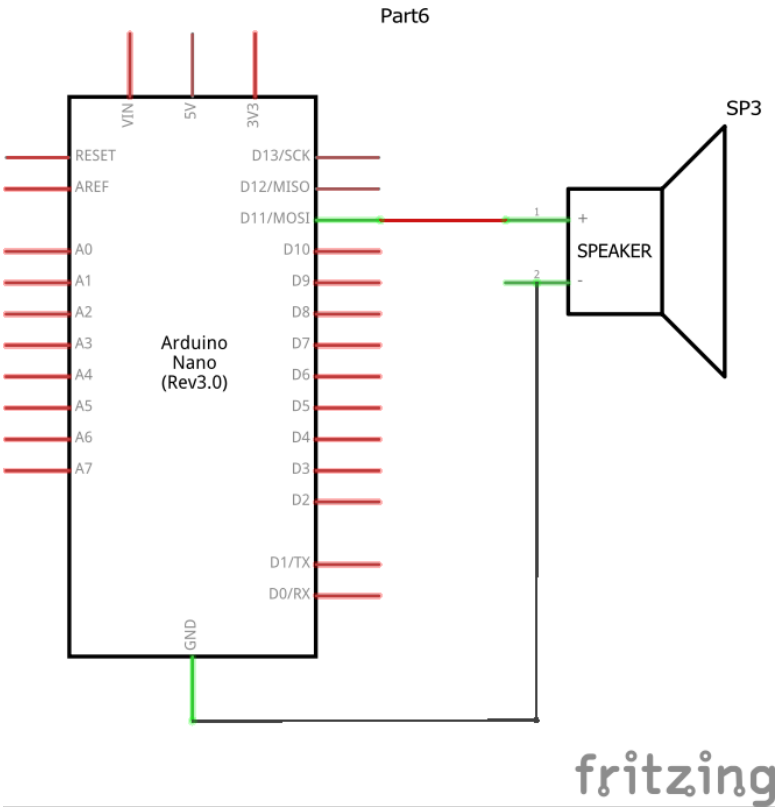


FIGURE 2.6 – Schéma de montage de bipeur avec l'Arduino

On doit aussi déclarer des mélodie qui seront joue par le bipeur, puisque nous suivons un modèle de programmation OOP, on va déclarer un Classe `Melody`, est une liste des mélodies disponibles. Ces mélodies seront jouées lors de :

— Une clique de bouton (Ça diffère d'un type de clique a l'autre)

— La sonnerie de la canne

— Le démarrage de la canne

```
#include "./pitches.h" // Maps each note to its frequency

class Ringtone {
public:
    static const int MAX_LENGTH = 100;
    const int *notes;
    int length;
    int duration;
    int sleep;

    Ringtone(const int *_notes, int _length, int _duration, int _delay) {
        notes = _notes;
```

```
13          length = _length;
14          duration = _duration;
15          sleep = _delay;
16      }
17
18      void play() {
19          for (int i = 0; i < length; i++) {
20              tone(11, notes[i], duration);
21              delay(sleep);
22          }
23      }
24  };
25
26  namespace Ringtones {
27      const int startupMelody[] = {NOTE_E6, NOTE_C7, NOTE_A6, NOTE_B6};
28      Ringtone startup(startupMelody, 4, 200, 150);
29
30      const int ringtoneMelody[] = {NOTE_A7, NOTE_D7};
31      Ringtone ringtone(ringtoneMelody, 2, 250, 150);
32
33      const int buttonClickMelody[] = {0, NOTE_E6, 0};
34      Ringtone buttonClick(buttonClickMelody, 3, 150, 100);
35
36      const int buttonDoubleClickMelody[] = {0, NOTE_F5, 0};
37      Ringtone buttonDoubleClick(buttonDoubleClickMelody, 3, 150, 100);
38
39      const int buttonLongClickMelody[] = {0, NOTE_CS6, 0};
40      Ringtone buttonLongClick(buttonLongClickMelody, 3, 150, 100);
41  }
```

Code 2.14 – Ringtone

2.3.3 Vocal feedback

Nous voulions inclure la rétroaction vocale dans la canne elle-même, mais malheureusement, les données vocales prennent trop d'espace, ce que Arduino n'a pas beaucoup d'espace (32 Ko), alors

nous avons décidé d'utiliser le téléphone comme un middleware pour jouer les phrases vocales.

Les instructions seront envoyées par la canne en utilisant l'instruction Bluetooth `SPEAK` comme décrit dans ce tableau 2.2

Screen readers (Lecteurs d'écran)

Un lecteur d'écran est une technique d'assistance, principalement utilisée par les personnes ayant une déficience visuelle. Il convertit du texte, des boutons, des images et d'autres éléments d'écran en discours ou en braille. Voyons ce qu'est un lecteur d'écran, comment il fonctionne et voyons les aveugles en action ! [15].

2.4 Les boutons de commande

Afin de commander le téléphone à distance, on aura besoin de quelque sort d'une interface de commande, d'où on a choisi d'utiliser 5 boutons poussoirs situés sur la canne.
Nous aurons 9 instructions qu'on doit traiter, d'où on aura besoin de trouver une méthode pour traiter 9 différents tâches, avec que 5 boutons.
Donc chaque boutons aura la possibilité d'exécuter deux différents tâches dépendant sur le type de la clique, d'une totalité de 10 tâches.

- Clique simple
- Clique lente

On a aussi ajouter un autre type, au cas on a voulu ajouter plus de tâches.

- Double clique

```
1  #include <Arduino.h>
2
3  #include "../../includes/ringtones.cpp"
4
5  class Button {
6  public:
7      // All times are in ms
8      unsigned long singlePressPolling = 1000;
9      unsigned long doublePressTimeout = 250;
10     unsigned long longPressTriggerTimeout = 1000;
11     unsigned long minPressPolling = 10;
```

```
12
13  private:
14      int _port;
15      bool _prevState = false;
16      bool _currentState = false;
17      bool _singlePressInQueue = false;
18      bool _longPressInProgress = false;
19      unsigned long _lastPressedAt = 0;
20
21      // Handlers
22      void (*_onPress)();
23      void (*_onDoublePress)();
24      void (*_onLongPress)();
25
26      static void _defaultHandler() {}
27
28  public:
29      Button(
30          int port,
31          void (*onPress)(),
32          void (*onDoublePress)() = _defaultHandler,
33          void (*onLongPress)() = _defaultHandler
34      ) {
35          _port = port;
36          _onPress = onPress;
37          _onDoublePress = onDoublePress;
38          _onLongPress = onLongPress;
39
40          pinMode(port, INPUT_PULLUP);
41      }
42
43      void listen() {
44          _currentState = !(bool)digitalRead(_port);
45
46          if (millis() < _lastPressedAt + minPressPolling)
47              return;
```

```
48
49          if (_singlePressInQueue) {
50              if (_currentState && _prevState) {
51                  if (millis() >= _lastPressedAt + longPressTriggerTimeout) {
52                      handleLongPress();
53                      _singlePressInQueue = false;
54                      _longPressInProgress = false;
55                      _lastPressedAt = millis();
56                  }
57              } else if (millis() <= _lastPressedAt + doublePressTimeout &&
                ↪  _currentState && !_prevState) {
58                  handleDoublePress();
59                  _singlePressInQueue = false;
60                  _lastPressedAt = millis();
61              } else if (millis() > _lastPressedAt + doublePressTimeout) {
62                  handlePress();
63                  _singlePressInQueue = false;
64                  _lastPressedAt = millis();
65              }
66          } else if (millis() >= _lastPressedAt + singlePressPolling &&
            ↪  _currentState && !_prevState) {
67              _singlePressInQueue = true;
68              _longPressInProgress = false;
69              _lastPressedAt = millis();
70          }
71
72          _prevState = _currentState;
73      }
74
75  private:
76      void handlePress() {
77          Ringtones::buttonClick.play();
78          _onPress();
79      }
80
81      void handleDoublePress() {
```

```
82          Ringtones::buttonDoubleClick.play();
83          _onDoublePress();
84      }
85
86      void handleLongPress() {
87          Ringtones::buttonLongClick.play();
88          _onLongPress();
89      }
90  };
```

Code 2.15 – Button

Bouton	Clique simple	Double clique	Clique lente
Milieu	-	-	Cliquer le bouton sélectionné
Gauche	Naviguer au bouton précédent	-	Envoyer l'SMS d'urgence
Droite	Naviguer au bouton suivant	-	Naviguer à la page "Explore"
Haut	Retourner à la page précédente	-	Redémarrer l'application
Bas	Cliquer le bouton sélectionné	-	Lancer la sonnerie du téléphone

TABLE 2.3 – Cartographie des boutons

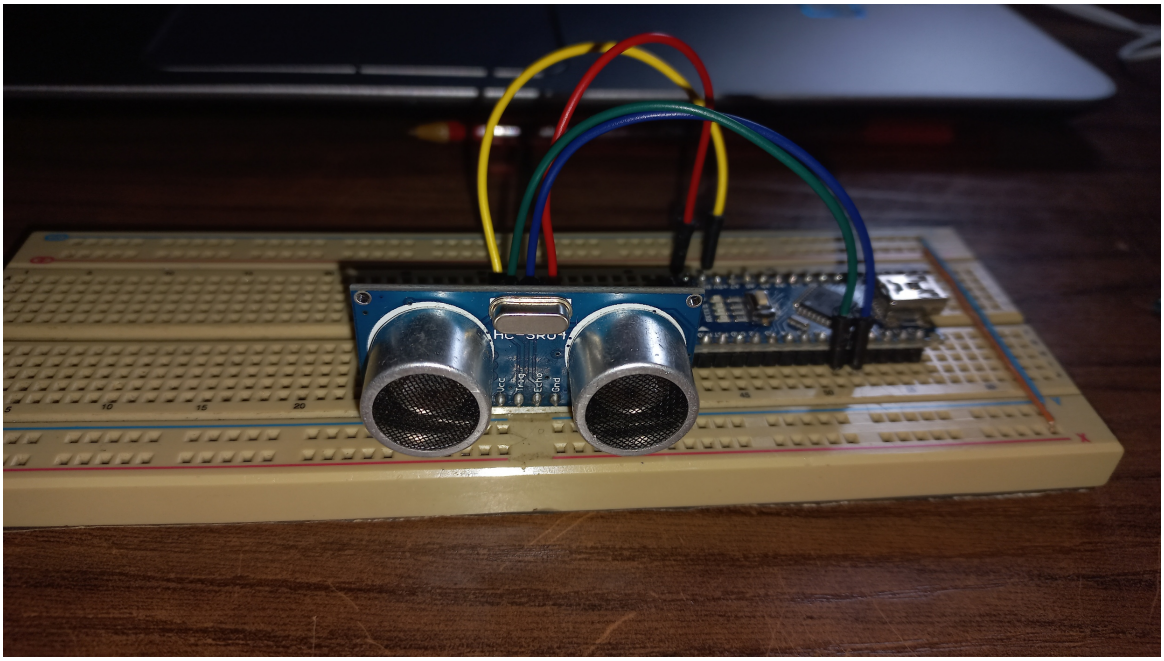
2.5 Les taches intelligents réalisés sur la canne

2.5.1 La détection d'obstacles

2.5.1.1 Montage

On connecte le capteur de proximité HC-SR04 avec l'arduino suivant le schéma suivant :

- VCC → 5V
- GND → GND
- TRIG → Broche 7
- ECHO → Broche 8



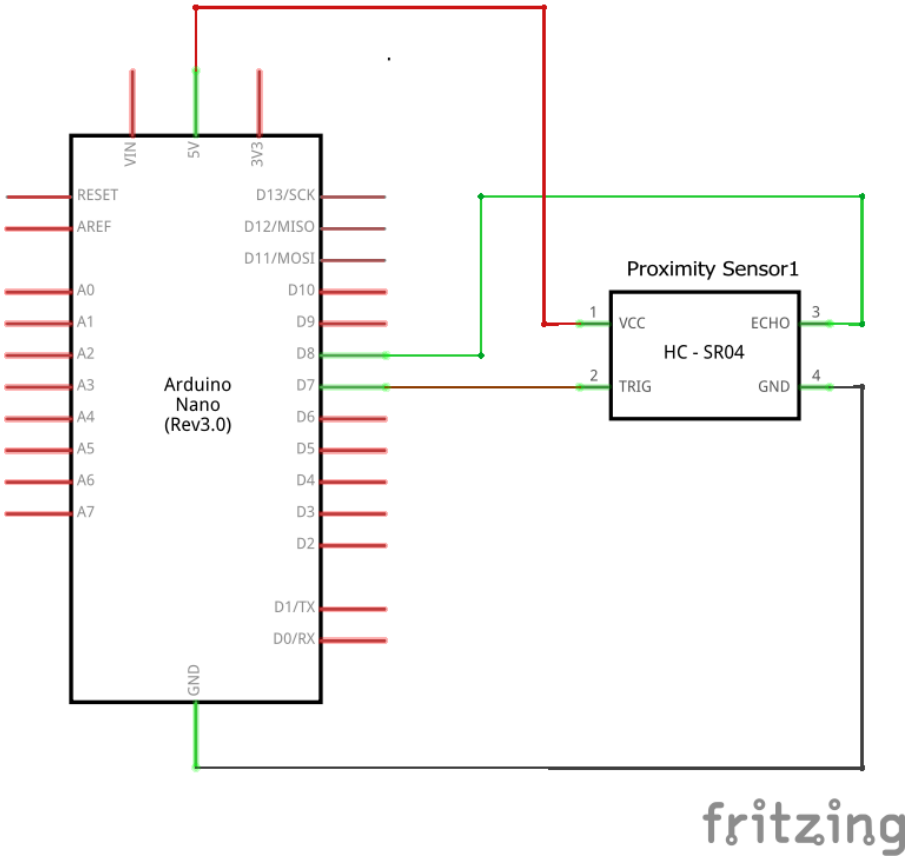


FIGURE 2.7 – Schéma et montage de la connexion entre HC-SR04 avec l'arduino

2.5.1.2 Fonctionnement

Comme expliqué dans cette paragraphe 1.3.1.1.3 Principe de fonctionnement du HC-SR04, on doit envoyer un signal d'état haut sur la broche Trigger d'une durée de 10µs, puis la broche Echo va indiquer le temps d'aller-retour pris par le son.

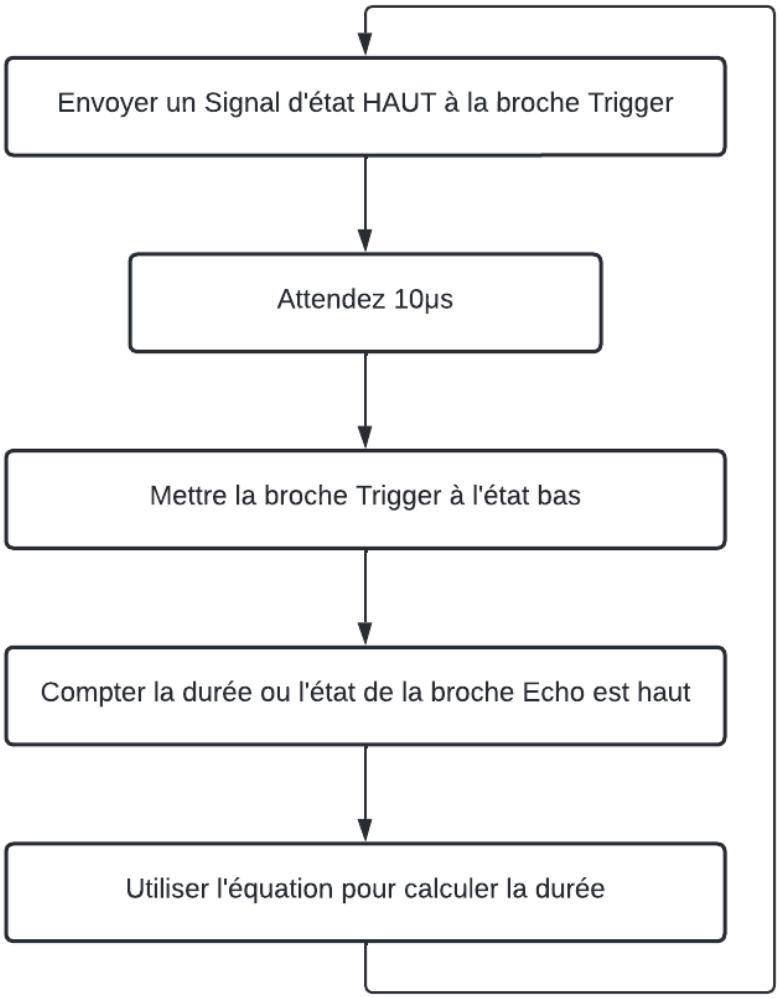
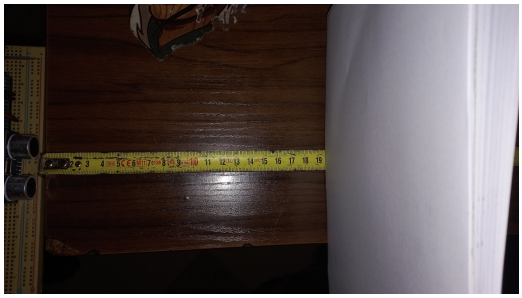
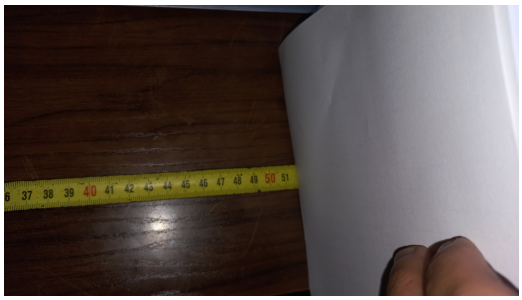


FIGURE 2.8 – Logigramme du code de HC-SR04

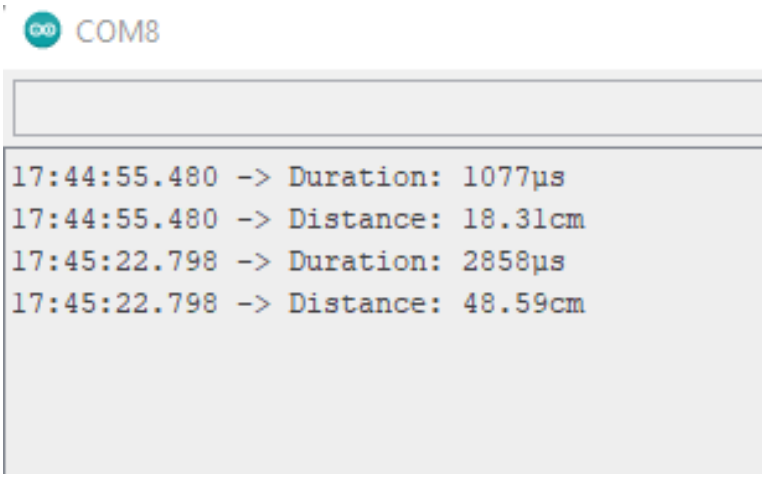
2.5.1.3 Des essais



(a) Distance de 19cm



(b) Distance de 52cm



(c) Le résultat des deux essais

FIGURE 2.9 – Essais pratique de HC-SR04

Vous pouvez remarquer que la distance lue par le HC-SR04 n'est pas exactement la distance indiquée sur le ruban à mesurer. Cela se résume à deux raisons :

— Nous avons peut-être mal aligné le capteur avec le ruban à mesurer, il y a donc une erreur de quelques millimètres.

— La vitesse du son dans l'air n'est pas constante, il y a des facteurs externes qui la modifient comme l'humidité et la température.

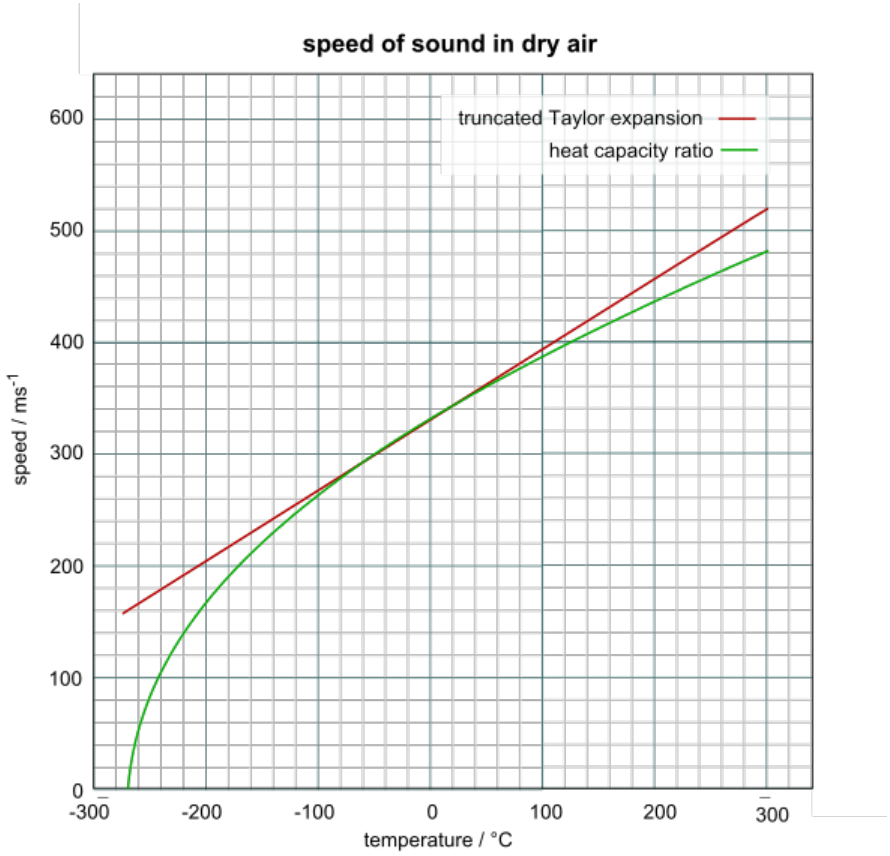


FIGURE 2.10 – Vitesse du son en fonction de la température

Pour notre projet, nous n'avons pas besoin que notre mesure soit trop précise, quelques centimètres d'erreur ne feront pas problème.

2.5.1.4 La vibration

Avant de parler de vibration, il faut d'abord parler de sortie analogique en Arduino.

Malheureusement, Arduino ne propose pas des broches de sortie analogiques, mais elle offre une autre solution qui est PWM. En un mot, PWM module la largeur de l'état HIGH dans un cycle de sortie, faisant le signal ressembler à un signal rectangulaire, ce qui fait moteurs et autres composants se comportent comme si le signal est une tension analogique changeante et changer leur vitesse de rotation.

$$v = A \cdot \frac{T_{on}}{T}$$

Arduino offre une range de 0. . .255 comme une sortie PWM, où

- $0 \rightarrow T_{on} = 0 \therefore v = 0$
- $255 \rightarrow T_{on} = T \therefore v = A = 5\text{volts}$

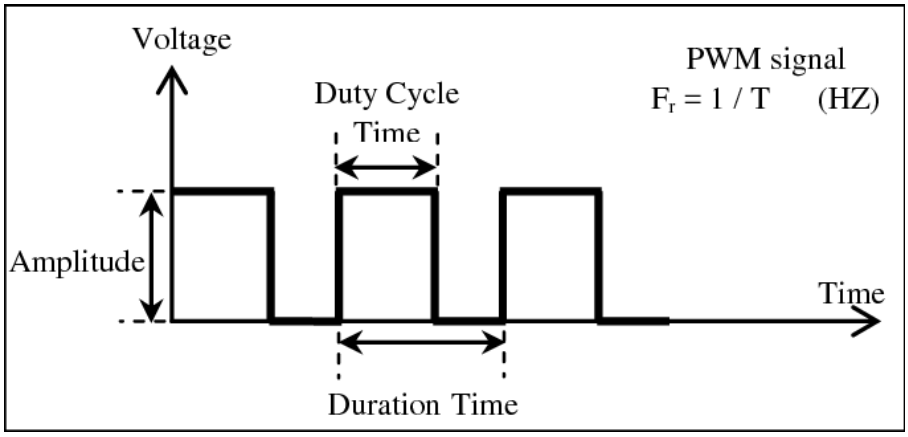


FIGURE 2.11 – Exemple d'un signal PWM

Après de nombreux tests, nous avons constaté que 1m est la distance parfaite pour alarmer l'utilisateur, où l'intensité des vibrations augmente plus l'obstacle est proche.

Où la sortie PWM passe de 50 pour des distances de 100cm à 255 pour toute distance inférieure à 2cm

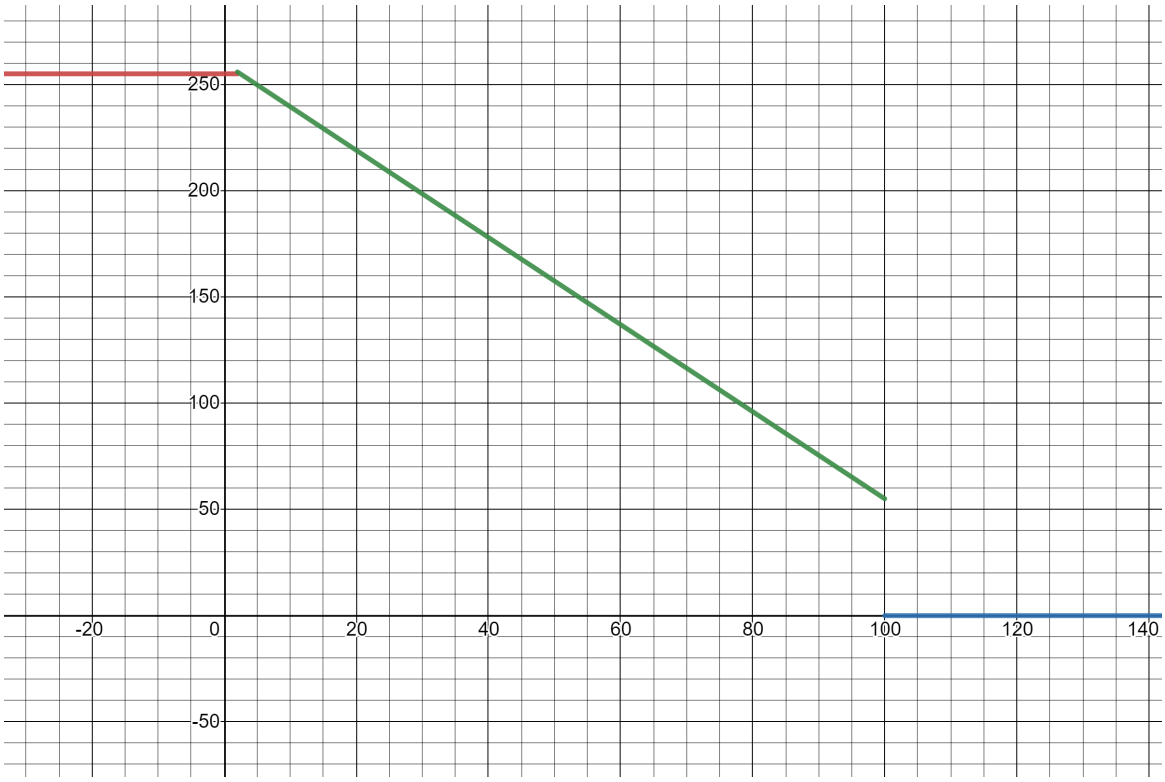


FIGURE 2.12 – Sortie PWM en fonction de la proximité d'obstacle

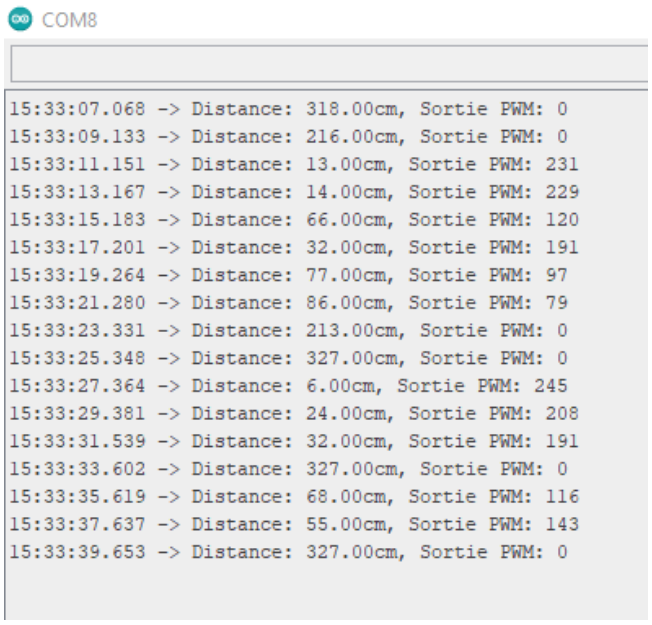


FIGURE 2.13 – Essais de la sortie PWM en fonction de la distance

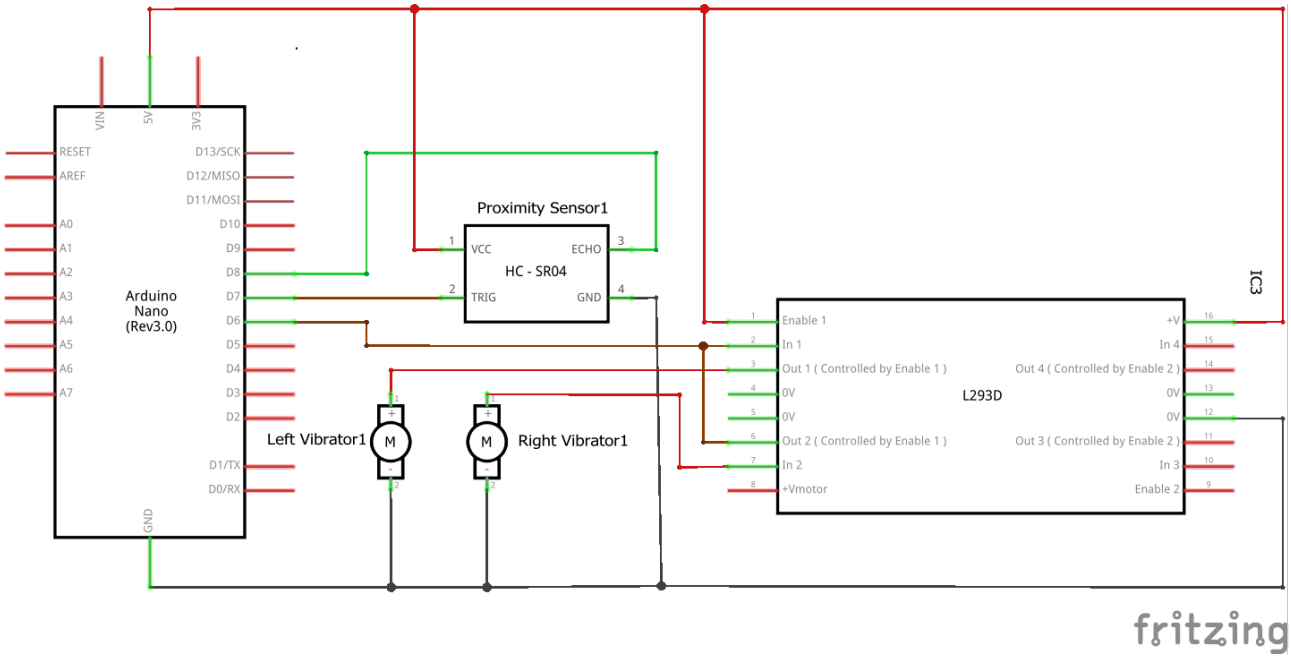


FIGURE 2.14 – Schéma avec la vibration de la connexion entre HC-SR04 avec l'arduino

On a utiliser la broche 6 car elle supporte la sortie Pulse Width Modulation (PWM).

2.5.1.5 Le problème avec l'approche actuelle

Bien que notre code semble bien fonctionner pour le moment, il causera un problème indésirable en cours de route.

Le problème avec l'approche actuelle est qu'il est "blocking-code", ce qui signifie que pendant qu'il est en train d'exécuter, aucune autre tâche peut être faite... et nous vérifions les obstacles en continu,

ce qui empêchent l'Arduino de faire autre chose que cela.

Le problème est causé par la fonction `pulseIn` intégrée dans l'Arduino, alors qu'il calcule la durée d'une impulsion, il bloque le cycle d'exécution.

Nous allons créer notre propre système de mesure, en utilisant la fonction `micros`.

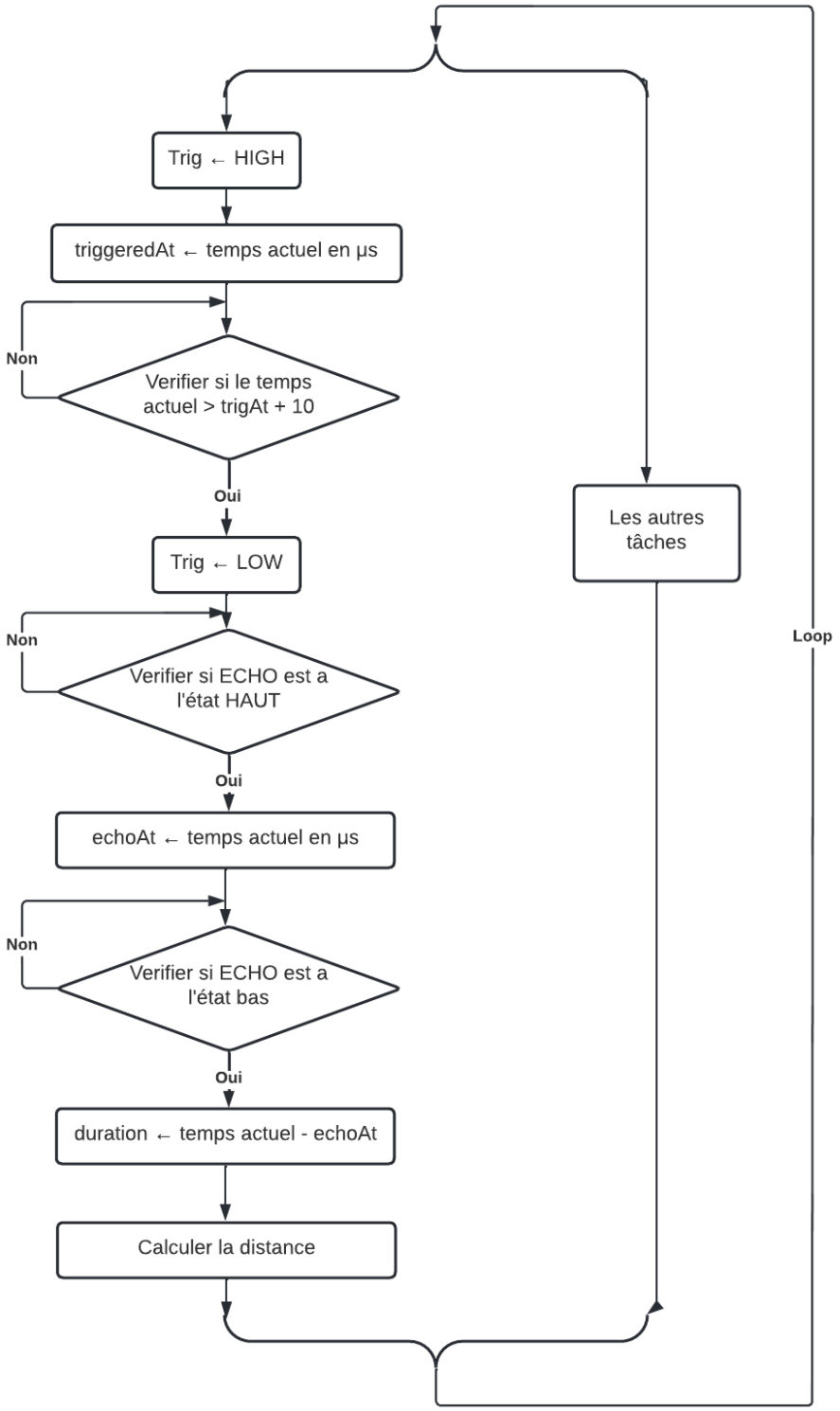


FIGURE 2.15 – Logigramme de notre principe personnalisé de fonctionnement de HC-SR04

Cette séquence fonctionne parfaitement, avec un seul autre problème : elle se bloque parfois. Nous pouvons corriger cela en ajoutant un chien de garde (WatchDog) qui réinitialise les propriétés enregistrées (echoAt, trigAt) pour redémarrer le cycle et le débloquer.

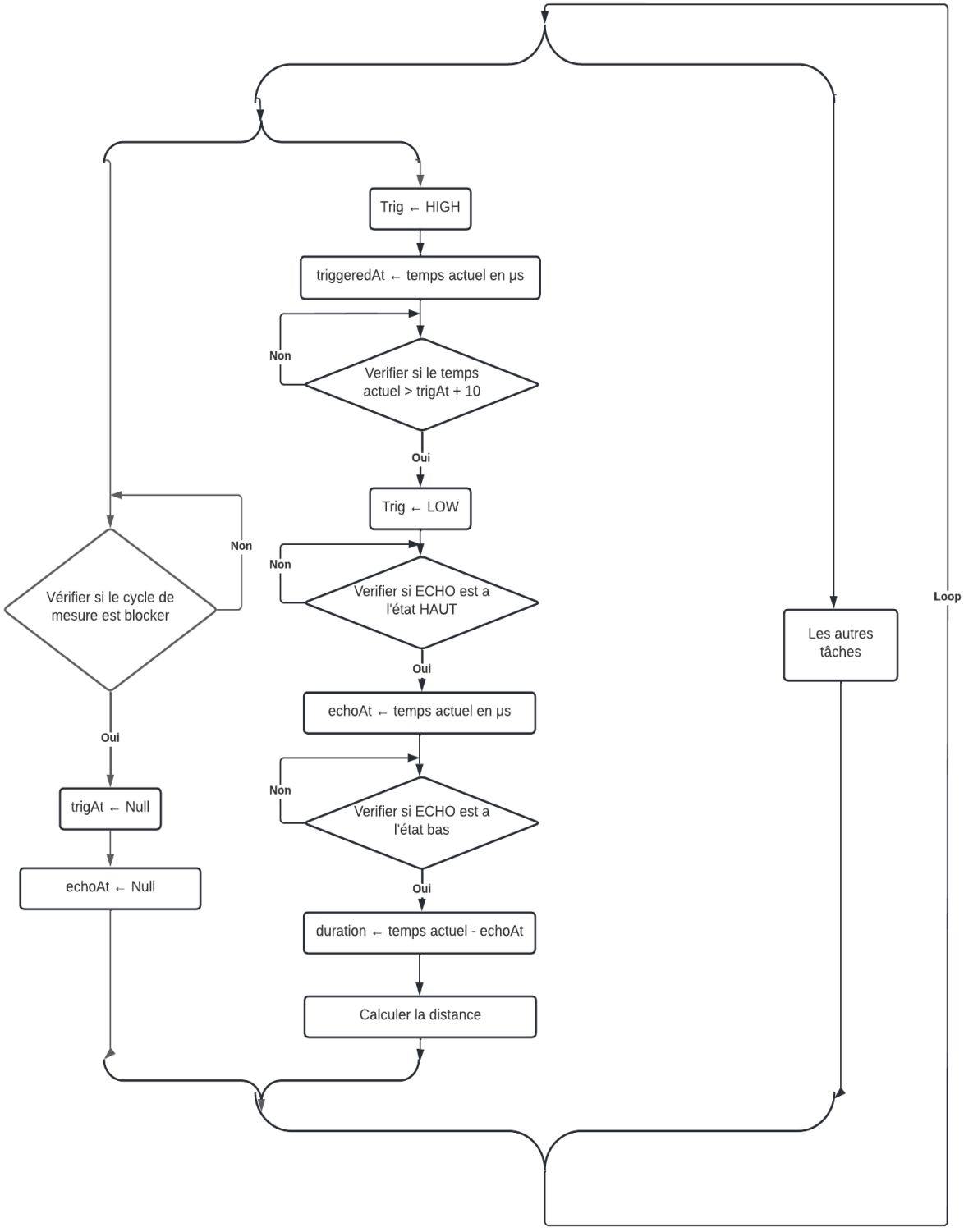


FIGURE 2.16 – Diagramme de notre principe personnalisé de fonctionnement de HC-SR04 avec le chien de garde

```cpp
#include <Arduino.h>

#include "../Watchdog.cpp"
class HC_SR04 {

public:
    double threshold = 0;      // in cm
    unsigned long polling = 0; // in ms
    Watchdog watchdog;

private:
    int _echoPin;
    int _trigPin;
    void (*_onUpdate)(double);
    bool _isTriggering = false;
    unsigned long _triggeringStartedAt = 0;
    bool _isEchoing = false;
    unsigned long _echoingStartedAt = 0;

public:
    // echo, trig
    HC_SR04(
        int echo,
        int trig,
        void (*onUpdate)(double)
    ) {
        _echoPin = echo;
        _trigPin = trig;
        _onUpdate = onUpdate;
        pinMode(_trigPin, OUTPUT);
        pinMode(_echoPin, INPUT);
        digitalWrite(_trigPin, LOW);
        watchdog.barkAfter = 500;
    }

```

```
    void listen() {
        watchdog.watch();
        if (watchdog.bark)
        {
            reset();
            watchdog.feed();
        }
        if (_isEchoing)
        {
            if (digitalRead(_echoPin) && !_echoingStartedAt)
            {
                _echoingStartedAt = micros();
            }
            if (!digitalRead(_echoPin) && _echoingStartedAt)
            {
                unsigned long _duration = micros() - _echoingStartedAt;
                double _distance = _duration * 0.034 / 2;
                _isEchoing = false;
                _echoingStartedAt = 0;
                watchdog.feed();
                _onUpdate(_distance);
            }
        }
        else
        {
            if (!_isTriggering)
            {
                digitalWrite(_trigPin, HIGH);
                _triggeringStartedAt = micros();
                _isTriggering = true;
            }
            else if (_isTriggering && micros() >= _triggeringStartedAt + 10)
            {
                digitalWrite(_trigPin, LOW);
                _triggeringStartedAt = 0;
                _isTriggering = false;
```

```
                _isEchoing = true;
            }
        }
    }

    void reset() {
        _isTriggering = false;
        _triggeringStartedAt = 0;
        _isEchoing = false;
        _echoingStartedAt = 0;
    }
};
```

Code 2.16 – HC-SR04

```
class Watchdog {
public:
    unsigned long long fedAt = 0;
    unsigned long long barkAfter = 1000;
    bool bark = false;

    Watchdog() {
        fedAt = millis();
    }

    void watch() {
        bark = millis() > (fedAt + barkAfter);
    }

    void feed() {
        fedAt = millis();
    }
};
```

Code 2.17 – WatchDog

```
namespace ProximityVibrator {
    const int _VibratorPort = 5;

    const int MIN_VIBRATION_VAL = 50;
    const int MAX_VIBRATION_VAL = 255;

    const int SENSOR_THRESHOLD = 100;

    void onUpdate(double distance) {
        int _val = max(MIN_VIBRATION_VAL + (SENSOR_THRESHOLD - distance) *
        ↪  (MAX_VIBRATION_VAL - MIN_VIBRATION_VAL) / (SENSOR_THRESHOLD - 2),
        ↪  0);
        if (_val < MIN_VIBRATION_VAL)
            _val = 0;
        analogWrite(_VibratorPort, _val);
    }

    HC_SR04 sensor(8, 7, onUpdate); // Echo, Trig, onUpdate

    void setup() {
        pinMode(_VibratorPort, OUTPUT);
        sensor.threshold = SENSOR_THRESHOLD;
    }

    void loop() {
        sensor.listen();
    }
}
```

Code 2.18 – Vibreur

2.5.1.6 Conclusion

Et avec cela, nous avons exécuté avec succès la tâche d'avertir l'utilisateur sur les obstacles devant lui, sans perturber d'autres tâches

2.5.2 La navigation

Comme discuté avant, nous utiliserons les différents services Google afin de trouver les informations à propos les différents endroits.

2.5.2.1 Les endroits à proximité

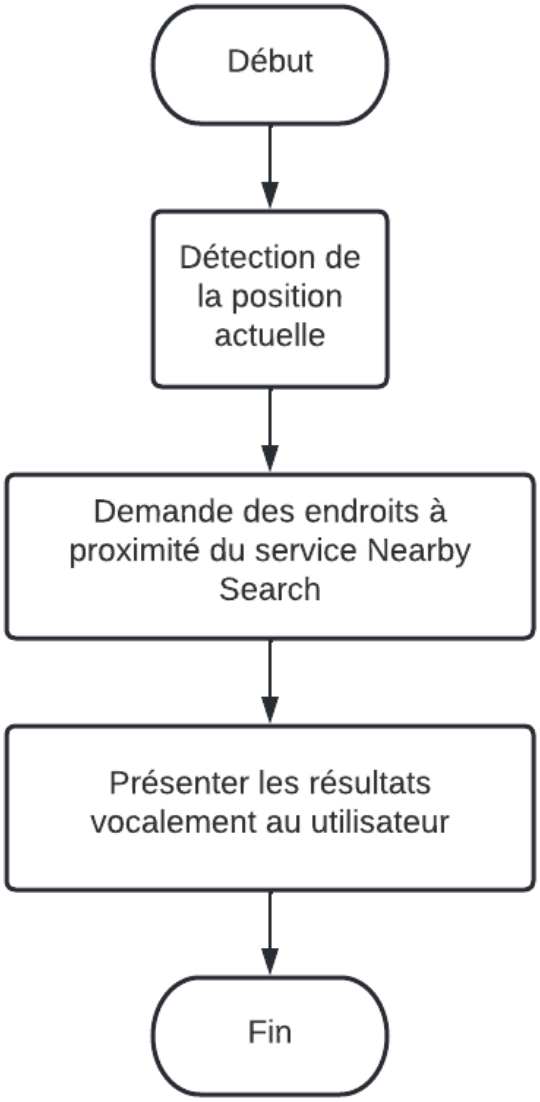
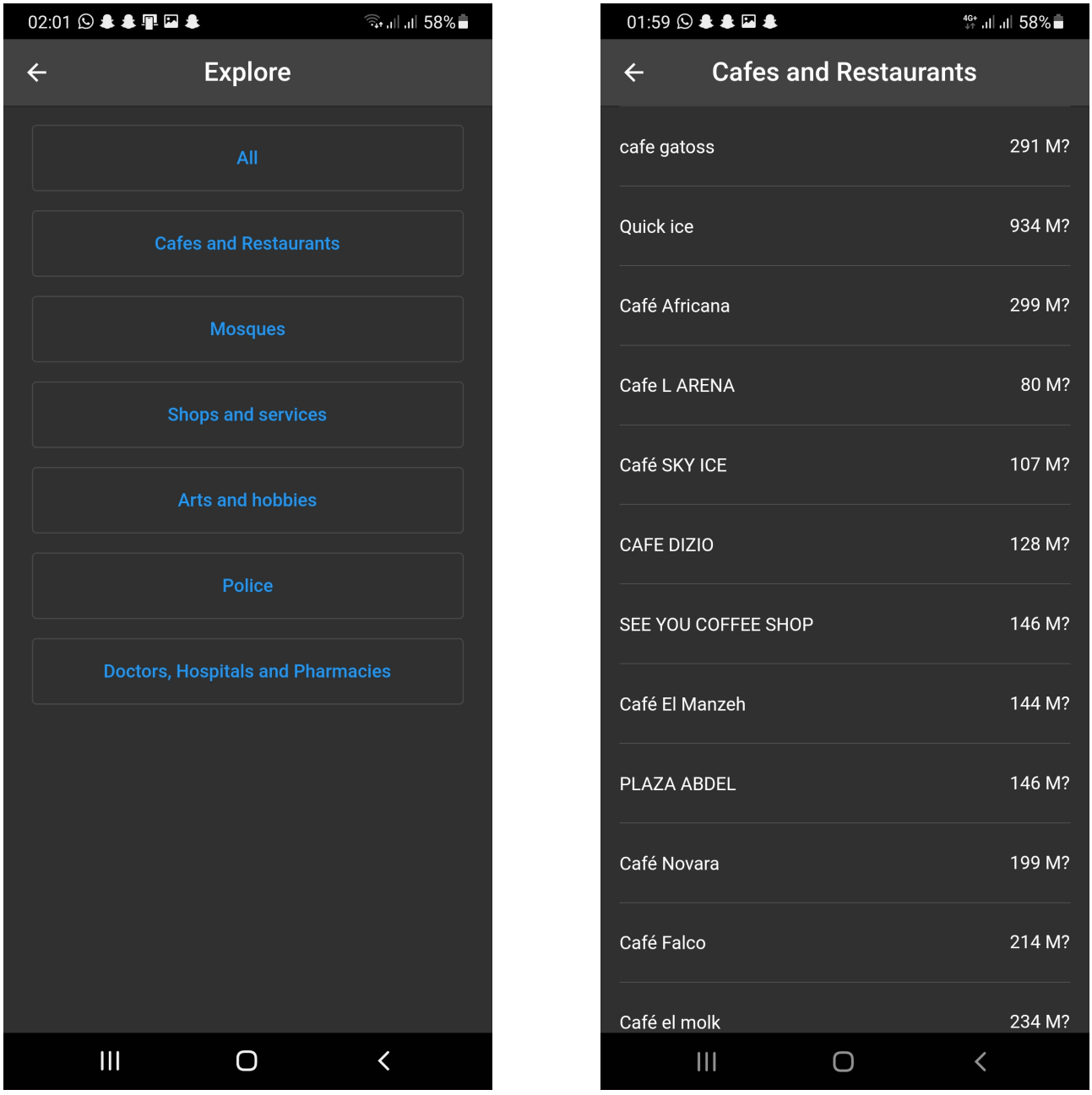


FIGURE 2.17 – Diagramme simplifié de la recherche des endroits à proximité

```
1  Future<List<SearchResult>?> searchNearbyPlaces(String? type) async {
2      var location =
       ↪  Provider.of<LocationService>(GlobalContextService.navigatorKey
3      .currentContext!, listen: false);
4      var currentLocation = await location.updateCurrentLocation();
```

```
5
6       if (currentLocation == null) {
7         return null;
8       }
9
10      try {
11        var result = await googlePlace.search.getNearBySearch(
12          Location(
13            lat: currentLocation.latitude,
14            lng: currentLocation.longitude,
15          ),
16          1000,
17          type: type,
18        );
19        return result?.results;
20      } catch (exception) {
21        inspect(exception);
22        return null;
23      }
24  }
```

Code 2.19 – Search for nearby places



(a) Sélection du type des endroits voulus

(b) Résultats de recherche

FIGURE 2.18 – Recherche des endroits à proximité

2.5.2.2 Recherche d'un endroit voulu

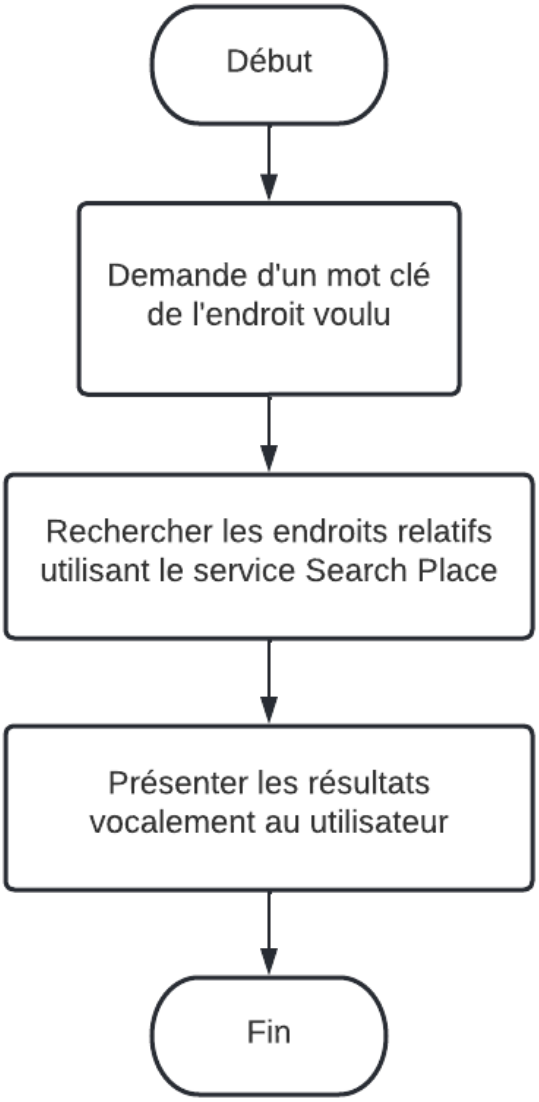
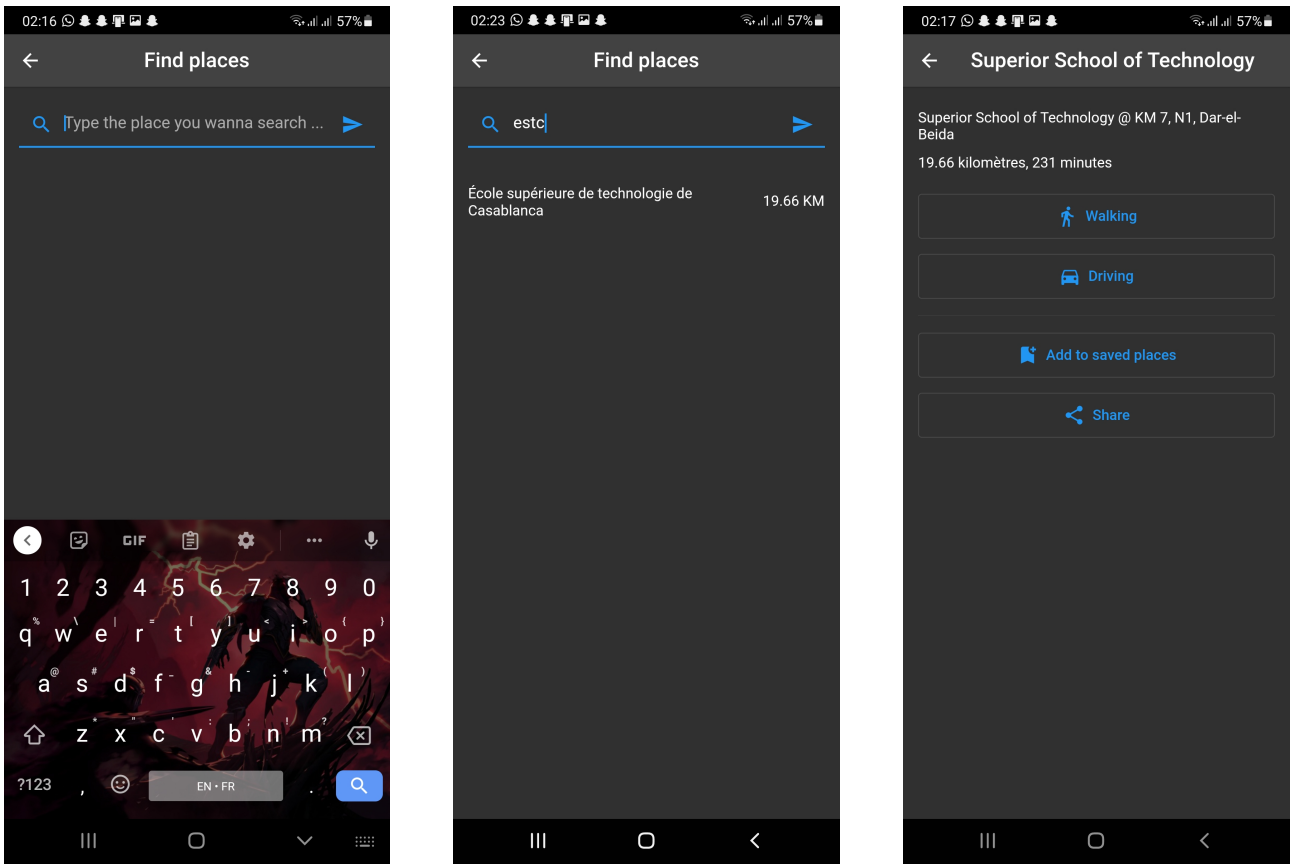


FIGURE 2.19 – Diagramme simplifié de la recherche d'un endroit voulu

```
void onSearch() async {
    setState(() {
      _places = [];
      isLoading = true;
    });

    var result = await googlePlace.search.getTextSearch(
      _inputController.value.text,
      language: "fr",
```

```
    );

    if (result == null || result.results == null) return;

    for (var element in result.results!) {
      _places.add(
        Place(
          info: DetailsResult(
            name: element.name,
            placeId: element.placeId,
            geometry: element.geometry,
          ),
        ),
      );

      setState(() {
        isLoading = false;
      });
    }

    for (var i = 0; i < _places.length; i++) {
      var place = _places.elementAt(i);
      _places[i].directions.walking = await
      ↪  Helpers.getDirections(destination: place);
      setState(() {});
    }
}
```

Code 2.20 – Recherche d'un endroit voulu

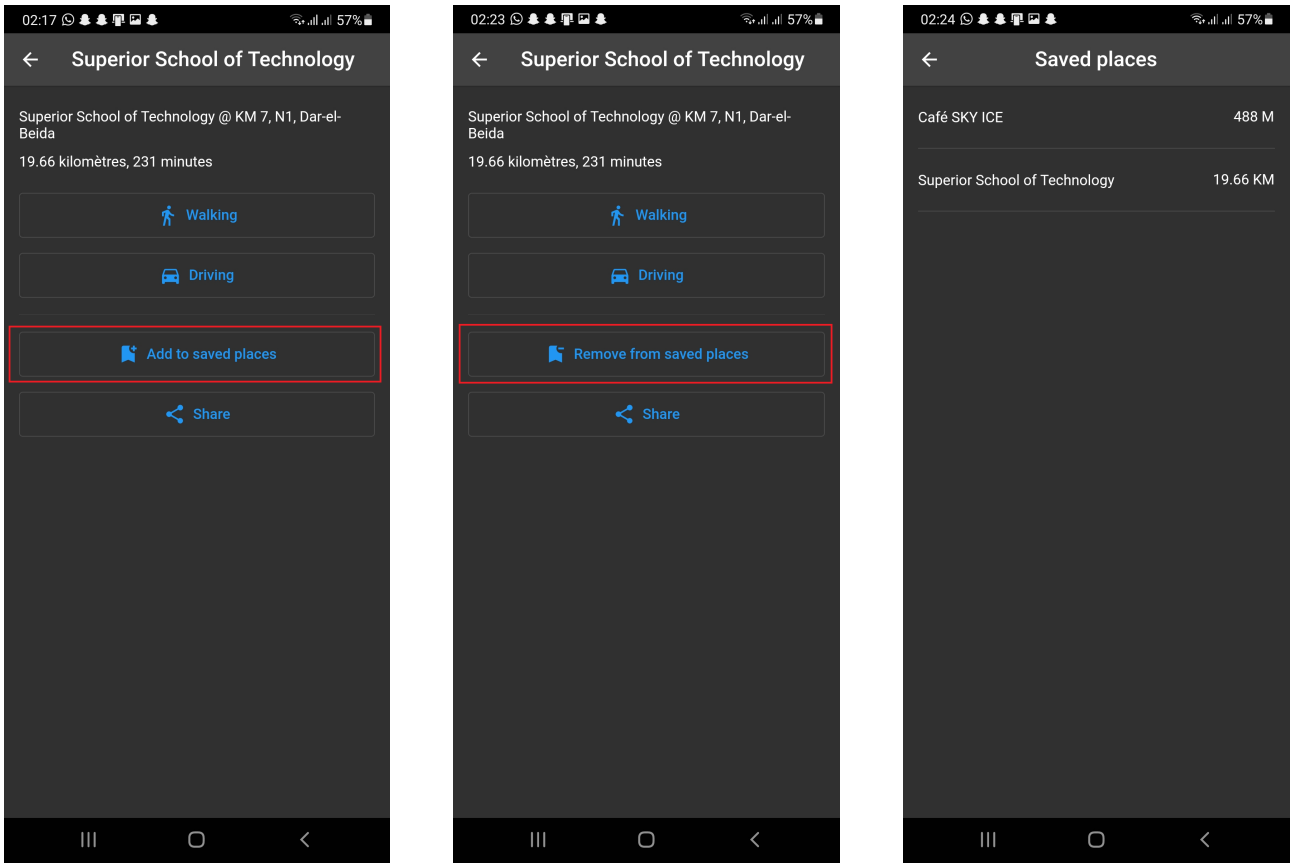


(a) Barre de recherche (b) Résultats de recherche (c) Endroit sélectionné

FIGURE 2.20 – Recherche d'un endroit voulu

2.5.2.3 Les endroits favoris

L'utilisateur peut ajouter un endroit à sa liste des endroits favoris afin de faciliter le retrouver



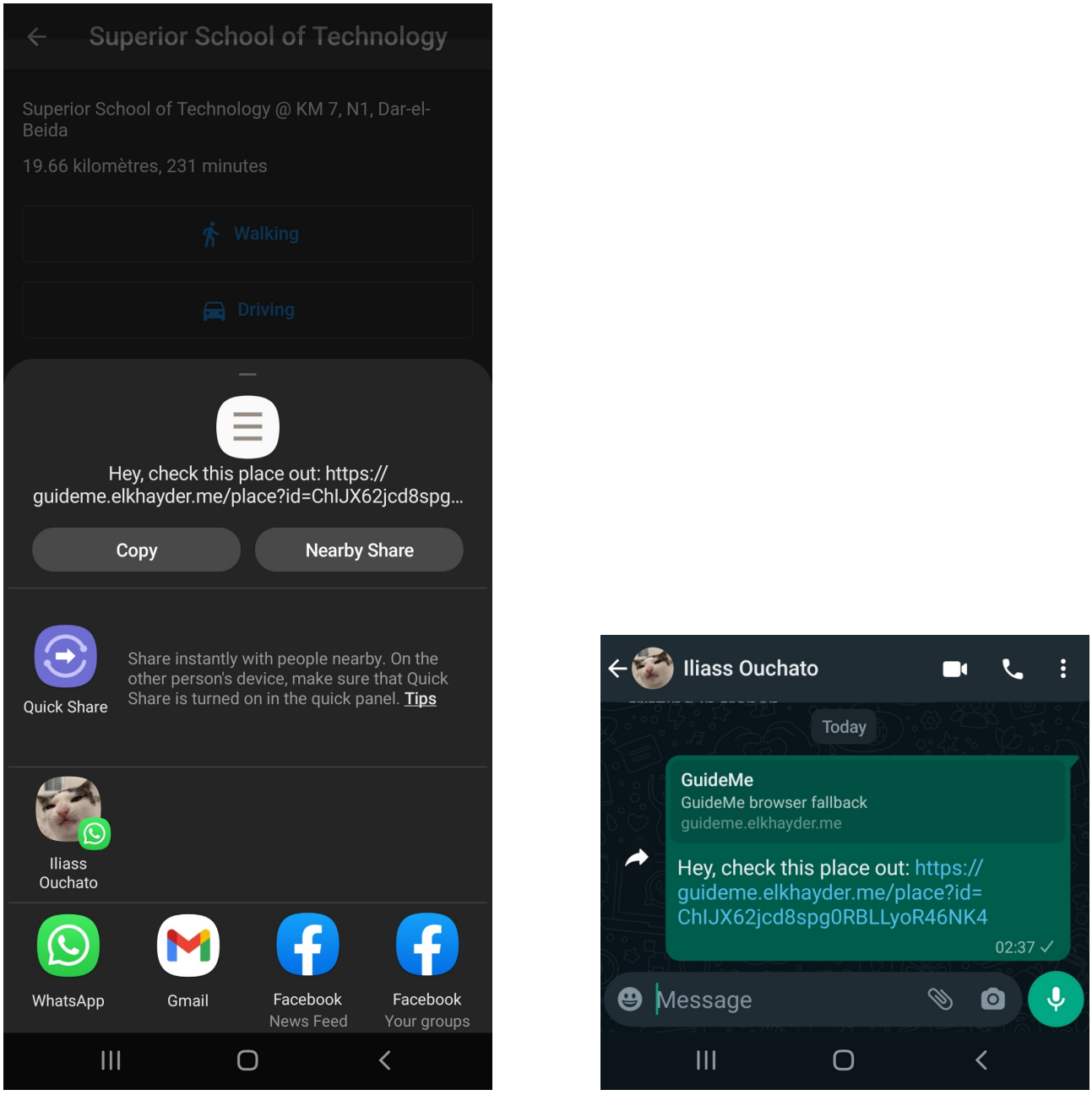
(a)

(b) Endroit ajouté a la liste des endroits favoris

(c) Liste des endroits favoris

FIGURE 2.21 – Les endroits favoris

2.5.2.4 Partage des endroits



(a) Les options disponibles pour partager le lien d'endroit

(b) Endroit partagé

FIGURE 2.22 – Partage des endroits

Le lien partagé ouvre l'application mobile sur la page correspondant à l'endroit, au cas si l'application n'est pas installée, ce lien va rediriger vers le site web créé dans le chapitre 2.6.

Le lient contient la valeur `place_id` qu'on a parlé dans le chapitre 2.1.3.3 Place Details, cette valeur sera utilisé pour trouver les informations de cet endroit.

2.5.2.5 La navigation

2.5.3 Localisation de la place en cas de perte

L'application téléphonique sera responsable d'obtenir la position actuelle de l'utilisateur à l'aide des capteurs GPS intégrés du téléphone.
Une fois que nous avons les coordonnées de l'utilisateur, nous récupérons les contacts d'urgence et le message d'urgence des paramètres. Puis Nous joignons un lien Google Maps à la latitude et la longitude de l'utilisateur avec le message d'urgence, et on l'envoie.

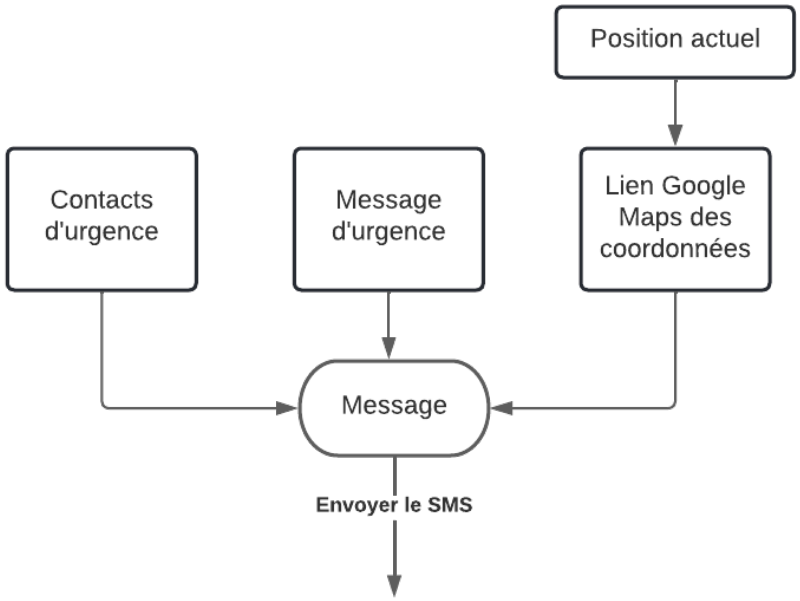


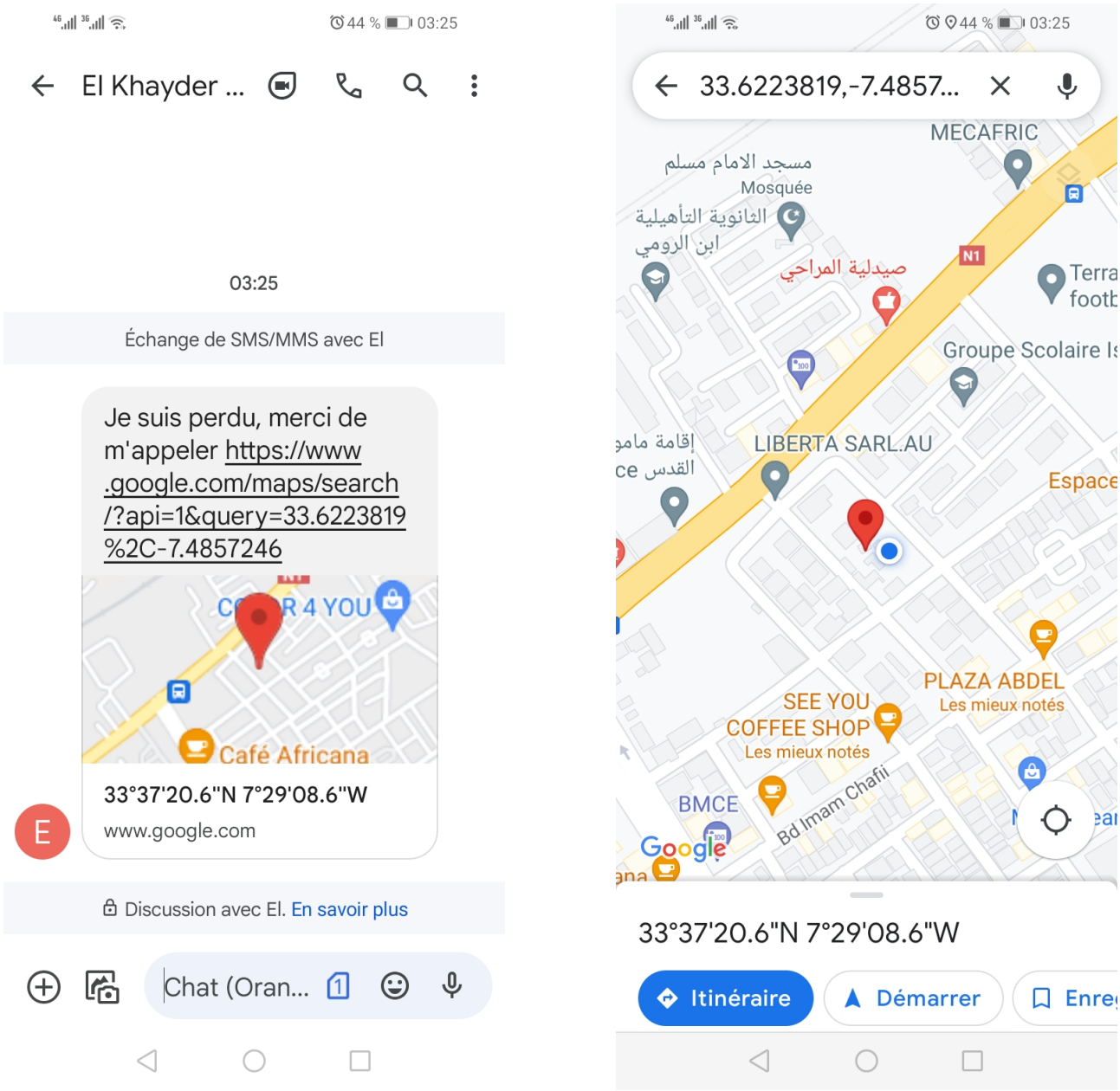
FIGURE 2.23 – Logigramme de l'SMS en cas de perte

```
class SendCurrentLocationSMS implements BluetoothPayloadHandler {
  final Telephony telephony = Telephony.instance;
  final LocationService location = LocationService();

  @override
  String command = "SEND_LOCATION_SMS";

```

```
  @override
  void handle(List<String> args) async {
    var currentLocation = await location.updateCurrentLocation();
    var prefs = await SharedPreferences.getInstance();
    String message = prefs.getString("emergencyMessage") ??
    ↪  Constants.defaultEmergencyMessage;
    List<EmergencyContact> contacts = await Helpers.fetchEmergencyContacts();

    if (currentLocation?.longitude != null && currentLocation?.latitude !=
    ↪  null) {
      message +=
      ↪  "https://www.google.com/maps/search/?api=1&query=${currentLocation
      !.latitude!.toString()}%2C${currentLocation.longitude!.toString()}";
    }

    if (contacts.isEmpty) {
      await Helpers.speak(
        "You don't have any emergency contacts, please add at least one in
        ↪  the settings menu.",
      );
      return;
    }

    for (var c in contacts) {
      await telephony.sendSms(
        to: c.phone,
        message: message,
      );
    }

    await Helpers.speak(
      "Location SMS sent to ${contacts.length} ${contacts.length == 1 ?
      ↪  "person" : "people"}",
    );
  }
```

```
38 }
```

Code 2.21 – Expéditeur d'SMS d'urgence



(a) SMS envoyé

(b) Position partagée

FIGURE 2.24 – SMS de position actuelle en cas de perte

On peut déclencher cette action à partir du téléphone directement, ou par les boutons de commande sur la canne à travers la communication Bluetooth entre les deux.

2.5.4 Trouver le téléphone avec la canne et vice-versa

En utilisant la canne, on peut lancer la sonnerie du téléphone pour que l'utilisateur peut le trouver facilement, comme on peut aussi lancer la sonnerie de la canne d'après le téléphone à condition que

les deux sont connecte entre eux avec Bluetooth

2.6 Le site web

Le site va fournir au destinataire les informations de l'endroit partagée au cas si l'application mobile GuideMe n'est pas installée sur son téléphone.

```
import React, { useEffect, useState } from "react";

type Place = {
   id: string;
   name: string;
   address: string;
   location: {
      lat: string;
      lng: string;
   };
};

const App = () => {
   const [place, setPlace] = useState<Place | null>(null);
   const [error, setError] = useState<String | null>(null);
   const [isLoading, setIsLoading] = useState(true);

   useEffect(() => {
      fetchPlaceInfos();
   }, []);

   const fetchPlaceInfos = async () => {
      const currentUrl = new URL(window.location.href);

      const apiEndpoint = new URL(
         "https://private-no-cors-proxy.herokuapp.com/https://maps.goog
         leapis.com/maps/api/place/details/json"
      );

```

```
        apiEndpoint.searchParams.append(
            "place_id",
            currentUrl.searchParams.get("id") ?? ""
        );

        apiEndpoint.searchParams.append(
            "key",
            "PRIVATE_KEY"
        );

        const request = await fetch(apiEndpoint.toString())
            .then((x) => x.json())
            .catch((e) => {
                setError(
                    "We encountered an error while loading place infos, try to
                    ↪ refresh the page."
                );
                return null;
            })
            .finally(() => setIsLoading(false));

        if (!request) return;

        if (request.status === "INVALID_REQUEST") {
            setError(
                "Seems like the link is broken, make sure you copied the exact
                ↪ one you received."
            );
            return;
        }

        setPlace({
            id: request.result.place_id,
            name: request.result.name,
            address: request.result.formatted_address,
            location: request.result.geometry.location,
```

```
64          });
65
66          document.title = `${request.result.name} - GuideMe`;
67      };
68
69      return (
70          <section id="head">
71              {isLoading && (
72                  <div className="lds-ring">
73                      <div />
74                      <div />
75                      <div />
76                      <div />
77                  </div>
78              )}
79              {error && <h3 className="error">{error}</h3>}
80              {place && (
81                  <>
82                      <h1 className="title">{place?.name}</h1>
83                      <h2 className="address">{place?.address}</h2>
84                      <div className="row">
85                          <a href={`https://guideme.elkhayder.me/?id=${place?.id}`}>
86                              <button className="purple">
87                                  <i className="fa-solid fa-location-crosshairs fa-2x"
                                    ↪  />
88                                  <span>Open in GuideMe</span>
89                              </button>
90                          </a>
91                          <a
92                              href="https://github.com/elkhayder/SmartCane-PFE-2022"
93                              target="_blank"
94                              rel="noreferrer"
95                          >
96                              <button className="blue">
97                                  <i className="fa-brands fa-android fa-2x" />
98                                  <span>Download app for Android</span>
```

```
                        </button>
                    </a>
                    <a
                        href={`https://maps.google.com?q=
                        ${place.location.lat},${place.location.lng}`}
                        target="_blank"
                        rel="noreferrer"
                    >
                        <button className="green">
                            <i className="fa-solid fa-map fa-2x" />
                            <span>Open in Maps</span>
                        </button>
                    </a>
                </div>
            </>
        )}
    </section>
  );
};

export default App;
```

Code 2.22 – Website

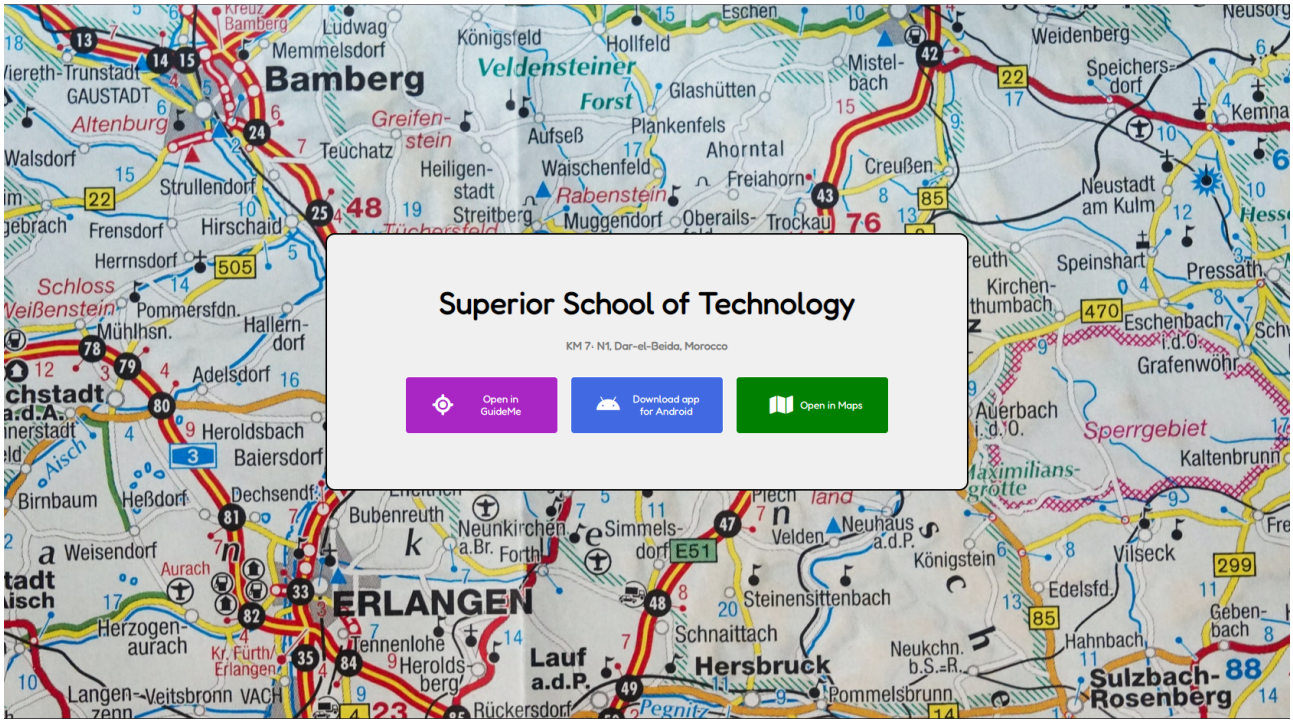


FIGURE 2.25 – Website : Version Web

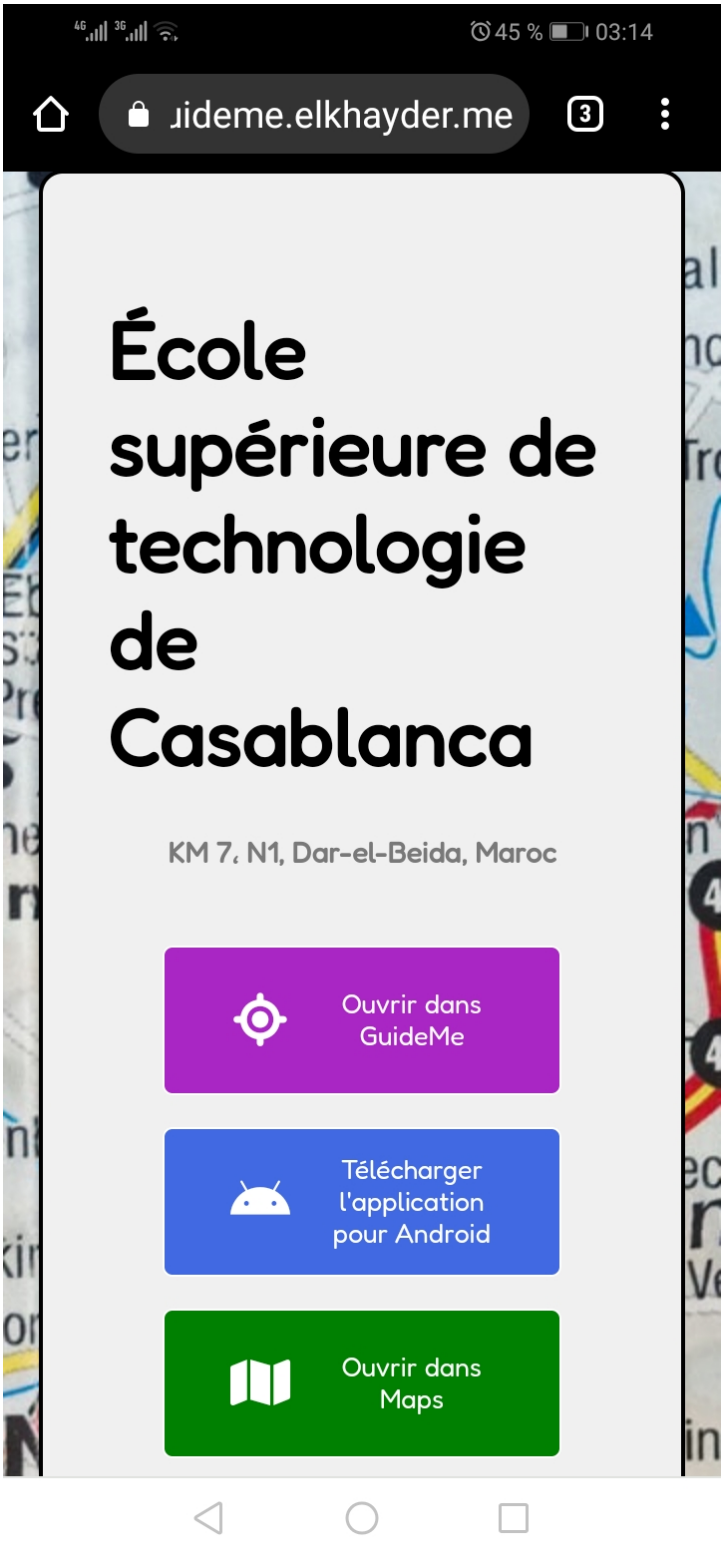


FIGURE 2.26 – Website : Version mobile

2.7 Conception de la canne

2.7.1 Introduction

Pour s'assurer que notre canne peut être facilement utilisé comme appareil portatif qui va accompagner l'utilisateur pendant ses promenades, qui va rendre sa vie plus facile plutôt que difficile en étant un fardeau, nous avons dû faire un design ergonomique et léger.

Pour satisfaire cela, les composants utilisés dans la fabrication de la canne ont dû travailler ensemble dans un boîtier compact c'est pourquoi nous avons conçu des circuits imprimés pour faire exactement cela et nous avons fait un boîtier qui va tenir ces PCBs d'une manière efficace pour économiser le maximum d'espace possible afin que nous puissions assurer un appareil portable qui sera fonctionnel et attrayant pour l'utilisateur en même temps.

2.7.2 Conception des circuits imprimés

2.7.2.1 Les circuits imprimés

Une carte de circuit imprimé (PCB) est une structure sandwich stratifiée de couches conductrices et isolantes. Les PCB ont deux fonctions complémentaires. La première consiste à apposer des composants électroniques aux endroits désignés sur les couches extérieures au moyen de la soudure. Le second est de fournir des connexions électriques fiables (et aussi des circuits ouverts fiables) entre les bornes du composant d'une manière contrôlée souvent appelée conception de circuits imprimés.

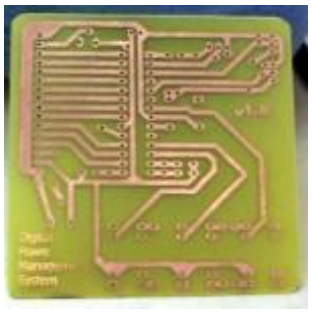


FIGURE 2.27 – Un exemple d'un PCB

2.7.2.2 Design

Pour porter nos composants électroniques à la vie dans la forme physique on fait la conception à l'aide d'un logiciel spécifique dédié pour la réalisation.

`Altium Designer` a été le premier logiciel de conception de circuits imprimés choisi, c'est un éditeur

complet pour les schémas électroniques utilisés par les professionnels et les ingénieurs de conception de circuits imprimés dans toutes sortes d'industries telles que les télécommunications.

FIGURE 2.28 – Interface du logiciel Altium Designer

Mais pendant que nous travaillions sur le logiciel nous avons rencontré tellement des difficultés en termes de facilité d'utilisation car c'était un environnement assez intense pour nous en tant que des étudiants qui n'ont pas beaucoup d'expérience dans la conception des PCBs, nous ne pouvions pas non plus nous permettre d'acheter une licence pour le logiciel alors nous avons utilisé une version craquée mais ce n'était pas stable il a continué à se planter et montrant des erreurs.

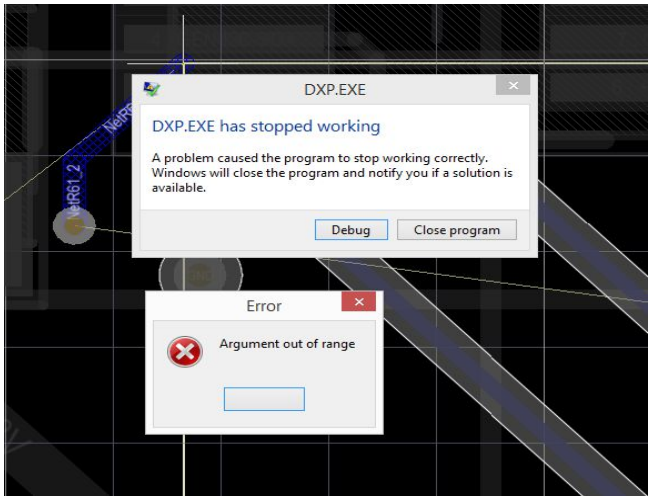


FIGURE 2.29 – Problème de plantage de Altium Designer

À d'autres moments, nous ne pouvions pas prendre le risque de perdre nos progrès, alors nous sommes passés à un autre logiciel que nous connaissons bien : Proteus : c'est un logiciel de conception de circuit imprimé et un simulateur de circuit qui est plus convivial pour les étudiants que Altium Designer qui est plus orienté pour les ingénieurs.

On veut réaliser deux circuits électroniques : un circuit pour les boutons et un autre pour les autres composants (Le circuit Arduino)

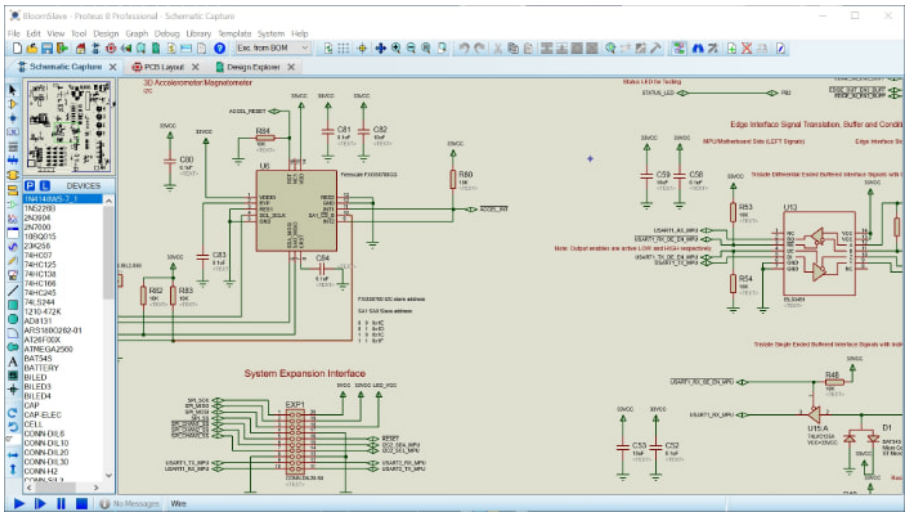


FIGURE 2.30 – Interface Proteus

2.7.2.3 Circuit arduino

Commençons par le circuit arduino : il y a eu beaucoup de changements qui se produisent tout au long de la réalisation ce circuit imprimé, donc nous allons commencer par le tout premier schéma :

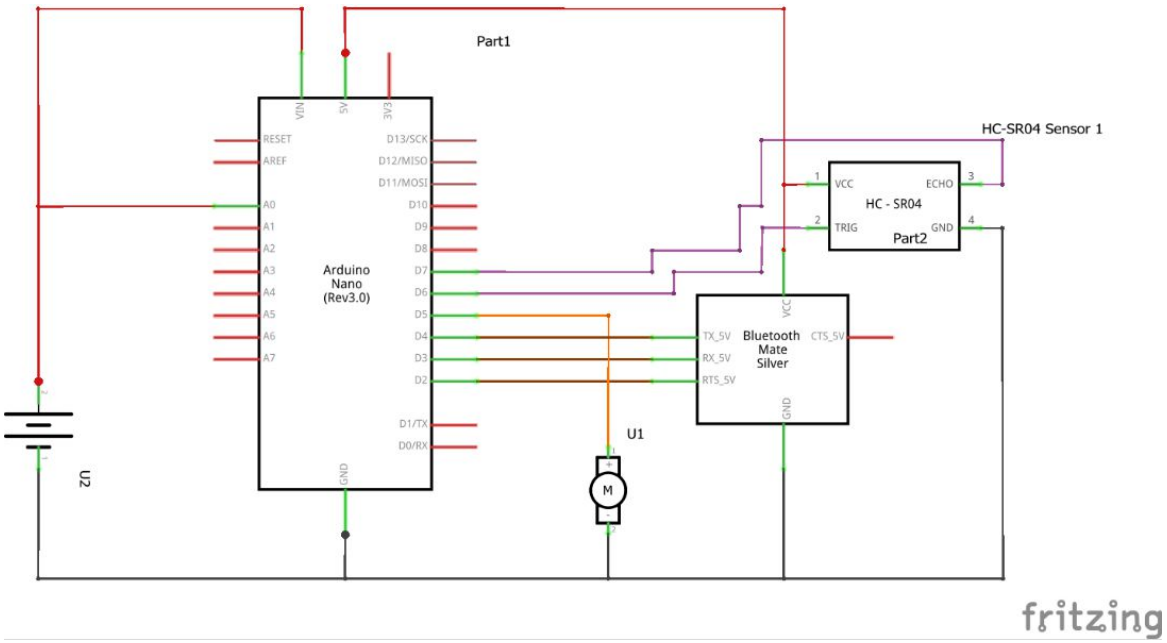


FIGURE 2.31 – Circuit d'Arduino

ce circuit a utilisé les principaux composants sur lesquels le projet était basé au début, nous avons travaillé sur le logiciel Fritzing pour avoir une idée générale de comment le circuit ressemblerait. Après nous avons réalisé le schéma ci-dessus sur Proteus en mode schematic capture.

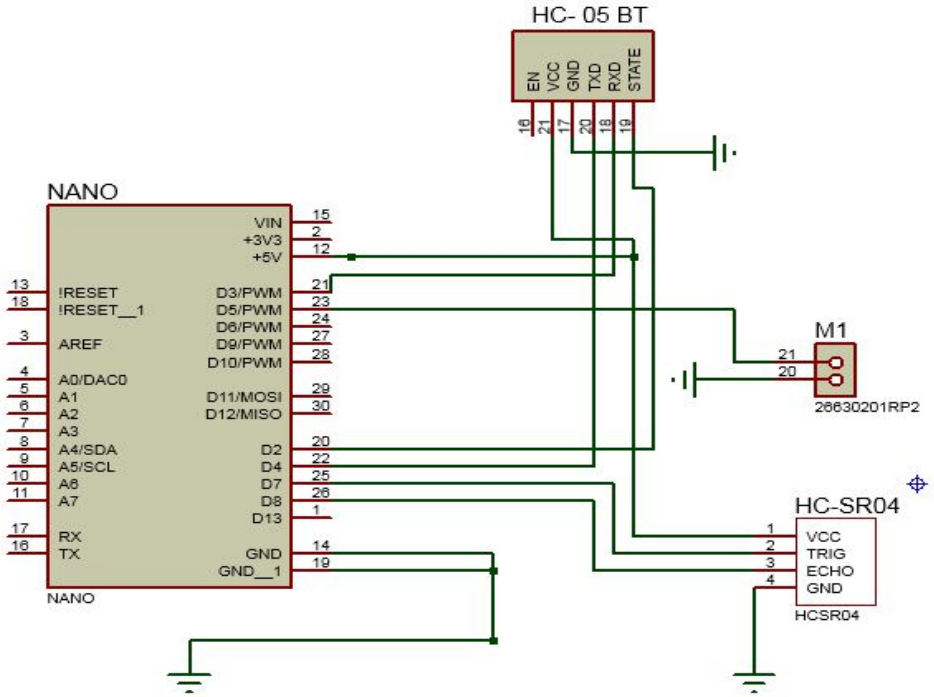
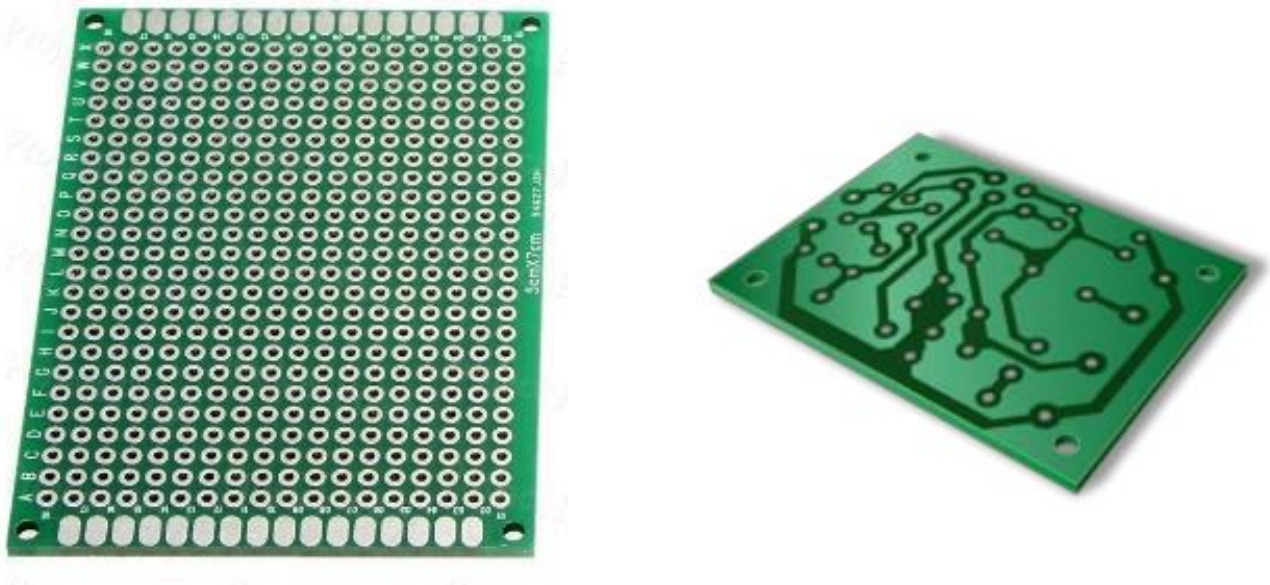


FIGURE 2.32 – Circuit Proteus d'Arduino

Après avoir fait le schéma, nous sommes passés à la conception d des circuits imprimés : Nous avons donc eu deux choix, soit celui d'utiliser le circuit imprimé pointillé traditionnel (figure a) ou le circuit imprimé (figure b). Nous avons donc choisi le deuxième choix simplement parce que c'est plus facile que le premier : il y a tellement de difficultés quand on travaille avec les circuits imprimés pointillés Ceux-ci relient les broches appropriées, évitant des mal connexions et etc.



(a) (b)

Sur Proteus nous passons au mode PCB layout et commençons à réaliser le circuit imprime en

plaçant les composants utilisés dans le mode schematic capture, en choisissant le bon package* pour chaque composant et en faisant finalement les connexions en cuivre ...

Ci-dessous vous trouverez chaque composant avec le package fourni trouvé sur les bibliothèques Proteus

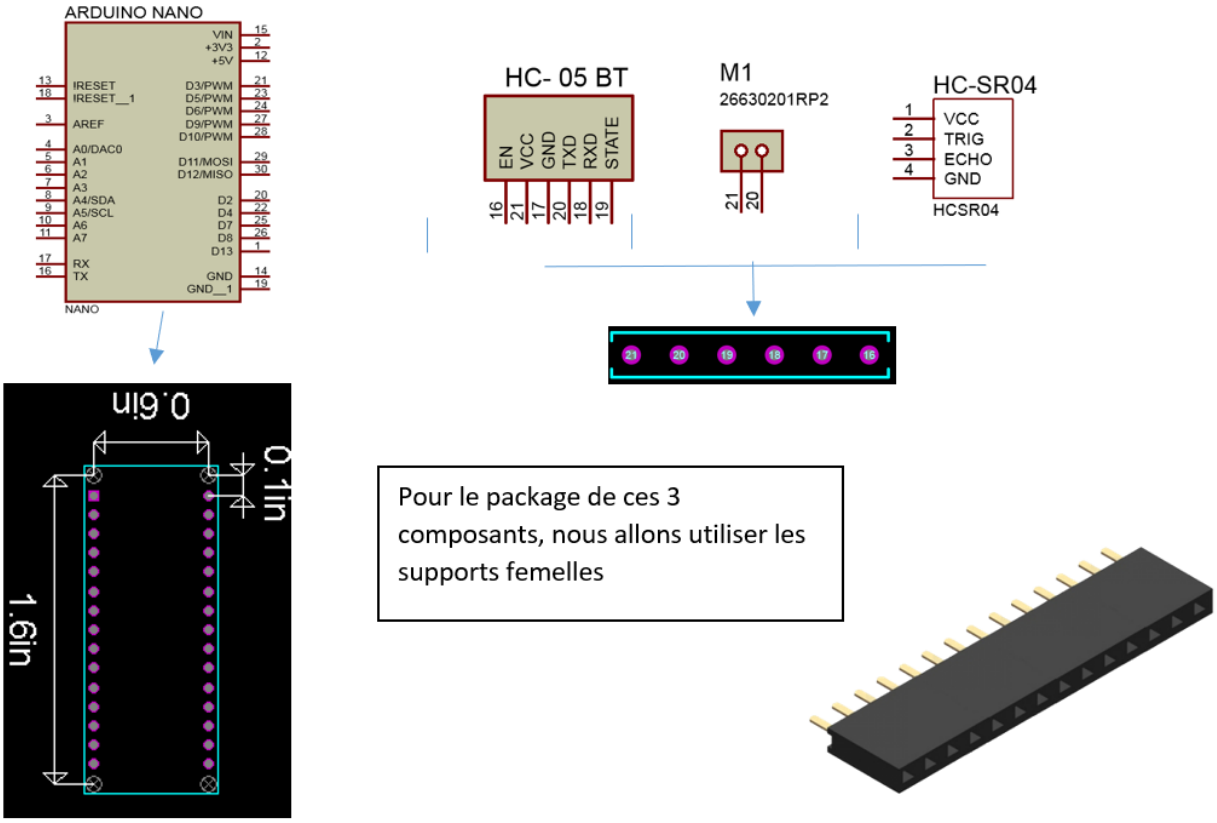


FIGURE 2.34 – Le package d'Arduino | Le support femelle

Nous avons eu un problème avec le package du support fourni par la bibliothèque Proteus, la distance entre deux broches est de 0.54mm plus grande que celle que nous allons utiliser, nous devons soit modifier le package sur Isis puisqu'ils ont de mauvaises mesures ou nous pouvons faire notre propre à partir de zéro.

Nous avons trouvé SnapeDA comme un site fiable d'où nous obtenons des packages des composants, mais nous n'avons pas pu trouver les mesures correctes pour les supports, nous avons donc décidé de les faire.

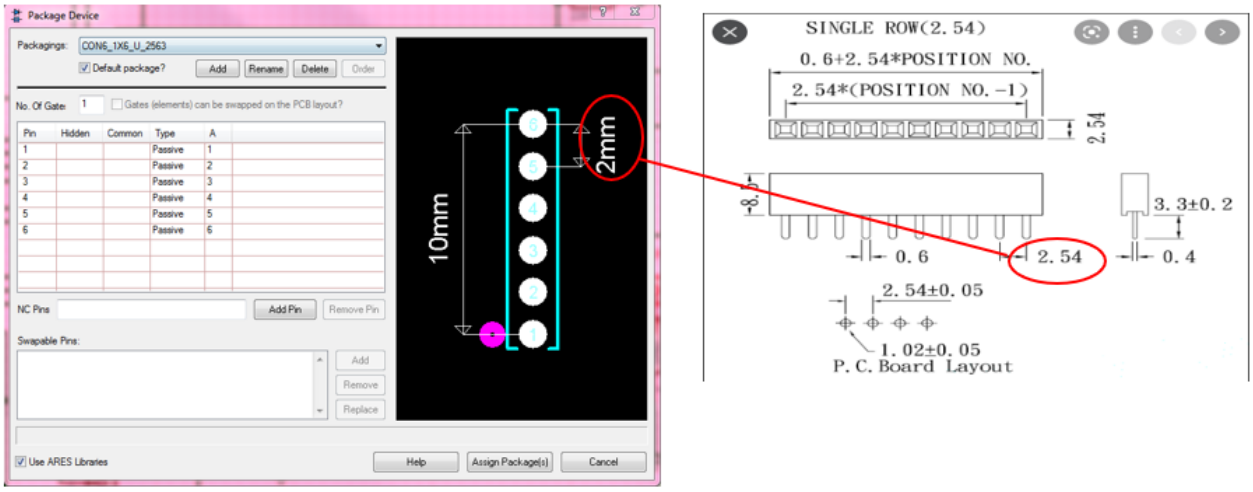


FIGURE 2.35 – Paramètres de support défaut de Proteus et les spécifications réelles

Nous nous sommes retrouvés avec cet circuit et nous avons également utilisé le 3D Visualizer mode pour avoir une version 3d proche de la vie réelle de la carte de sorte que nous puissions travailler sur elle comme une base de notre conception de boîtier

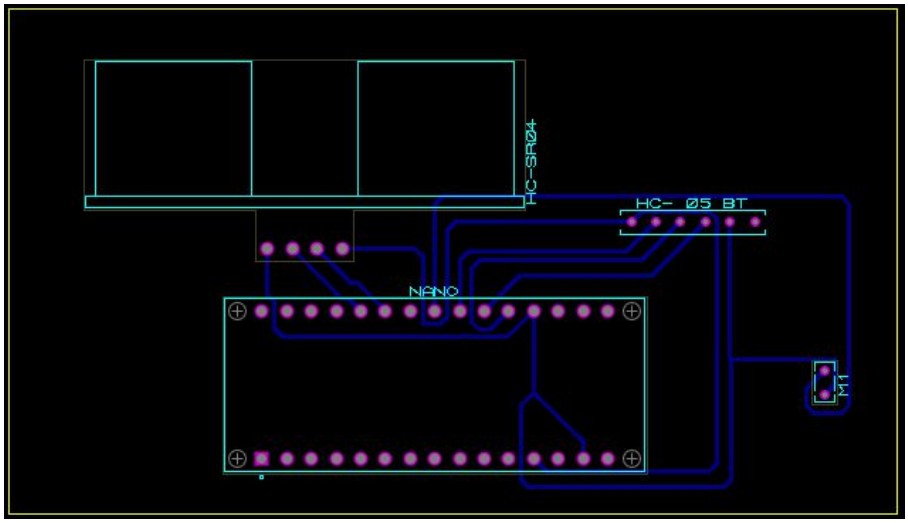


FIGURE 2.36 – Layout du PCB du circuit Arduino

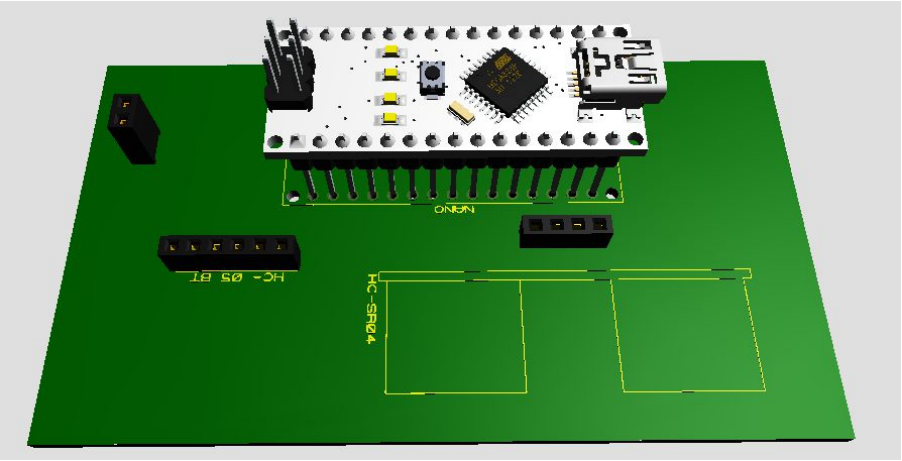


FIGURE 2.37 – Modèle 3D du PCB du circuit Arduino

Nous avons commencé à ajouter plus de composants dans le schéma : nous ne pouvions plus alimenter le moteur vibreur avec la broche d'Arduino car elle ne produit qu'un faible courant de 40mA et le moteur a besoin d'au moins 70ma de courant pour fonctionner correctement, également un de ces composants pourrait être endommagé en cours de route donc nous avons utilisé un moteur driver de référence l293d, nous avons aussi ajouté un moteur secondaire pour augmenter l'intensité de feedback

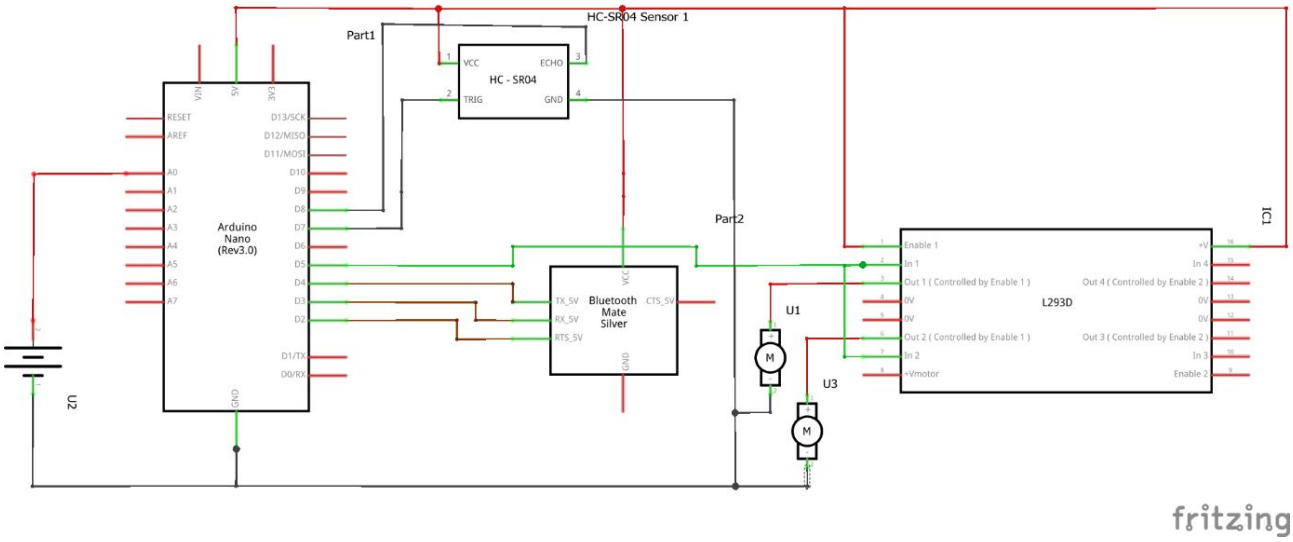


FIGURE 2.38 – Circuit d'Arduino finale

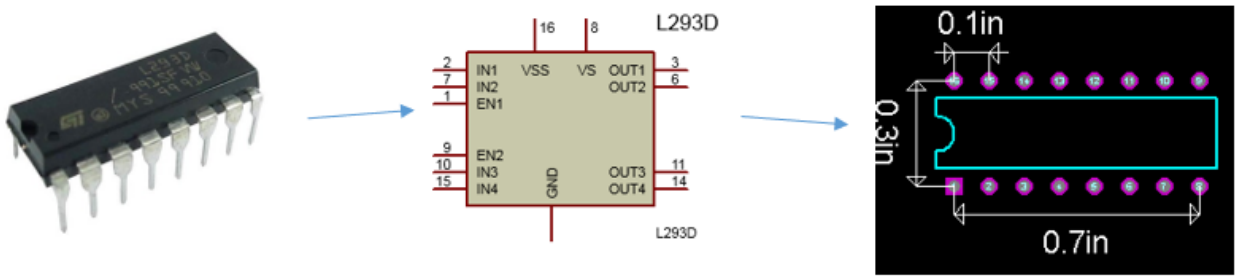


FIGURE 2.39 – Le L293D

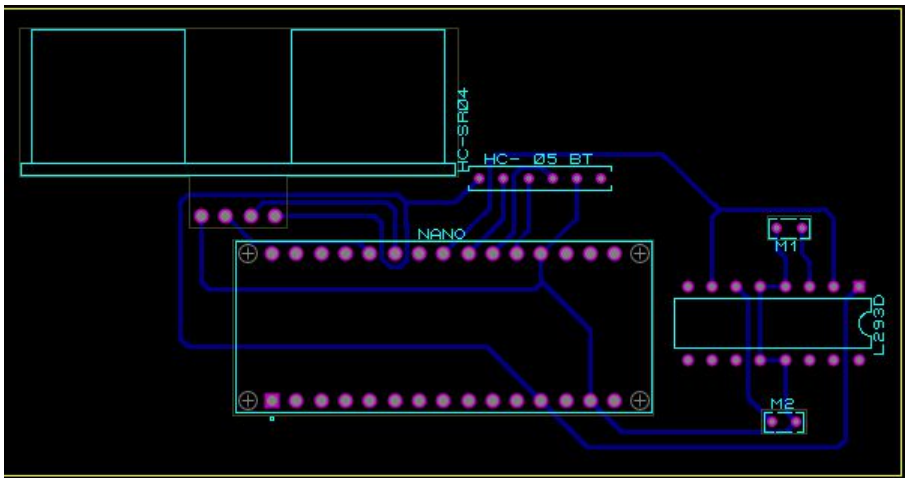


FIGURE 2.40 – Le PCB du circuit Arduino avec le L293D

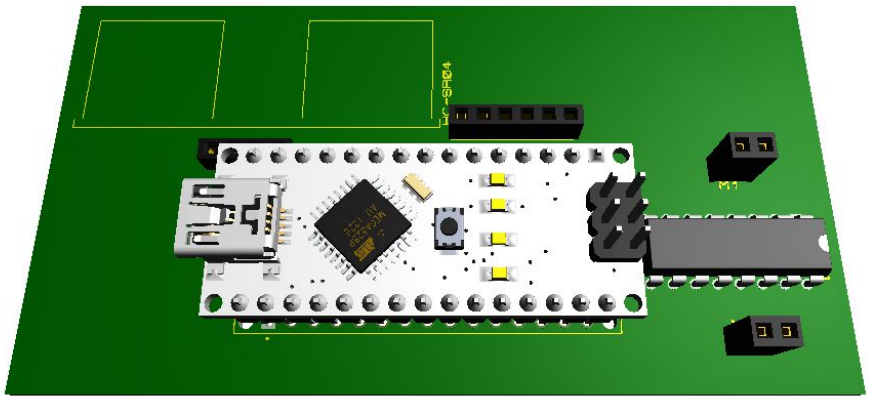
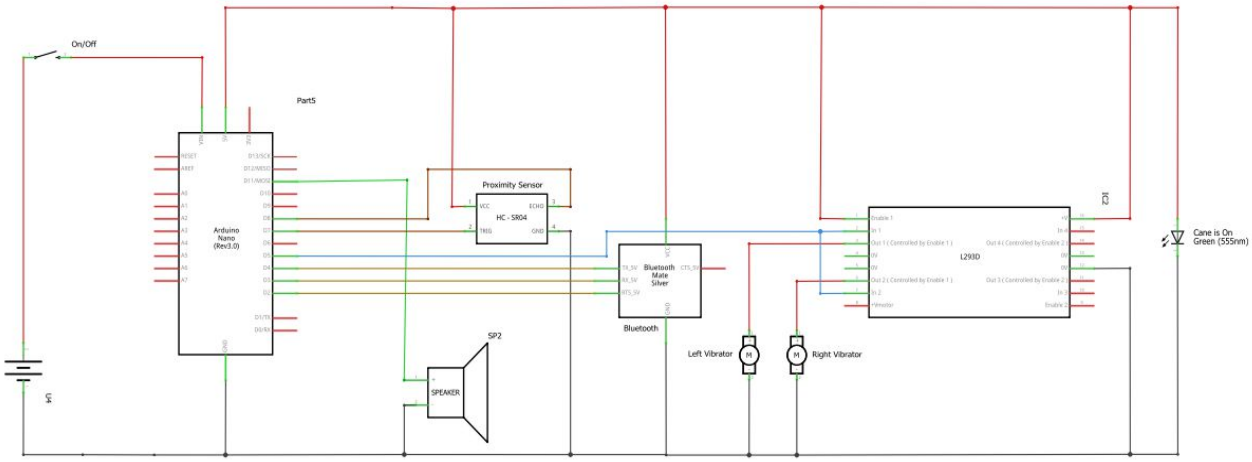


FIGURE 2.41 – Modèle 3D PCB du circuit Arduino avec le L293D

Et le dernier changement que nous avons fait concernant le circuit Arduino a été d'ajouter un buzzer, une lampe flash, et un interrupteur à glissière pour allumer/éteindre le circuit. Nous avons également ajouté un support femelle qui va être connecté au circuit des boutons



Tous les nouveaux composants ajoutés utilisent les mêmes supports femelles qui ont été utilisés précédemment. Nous nous retrouvons avec le circuit ci-dessous

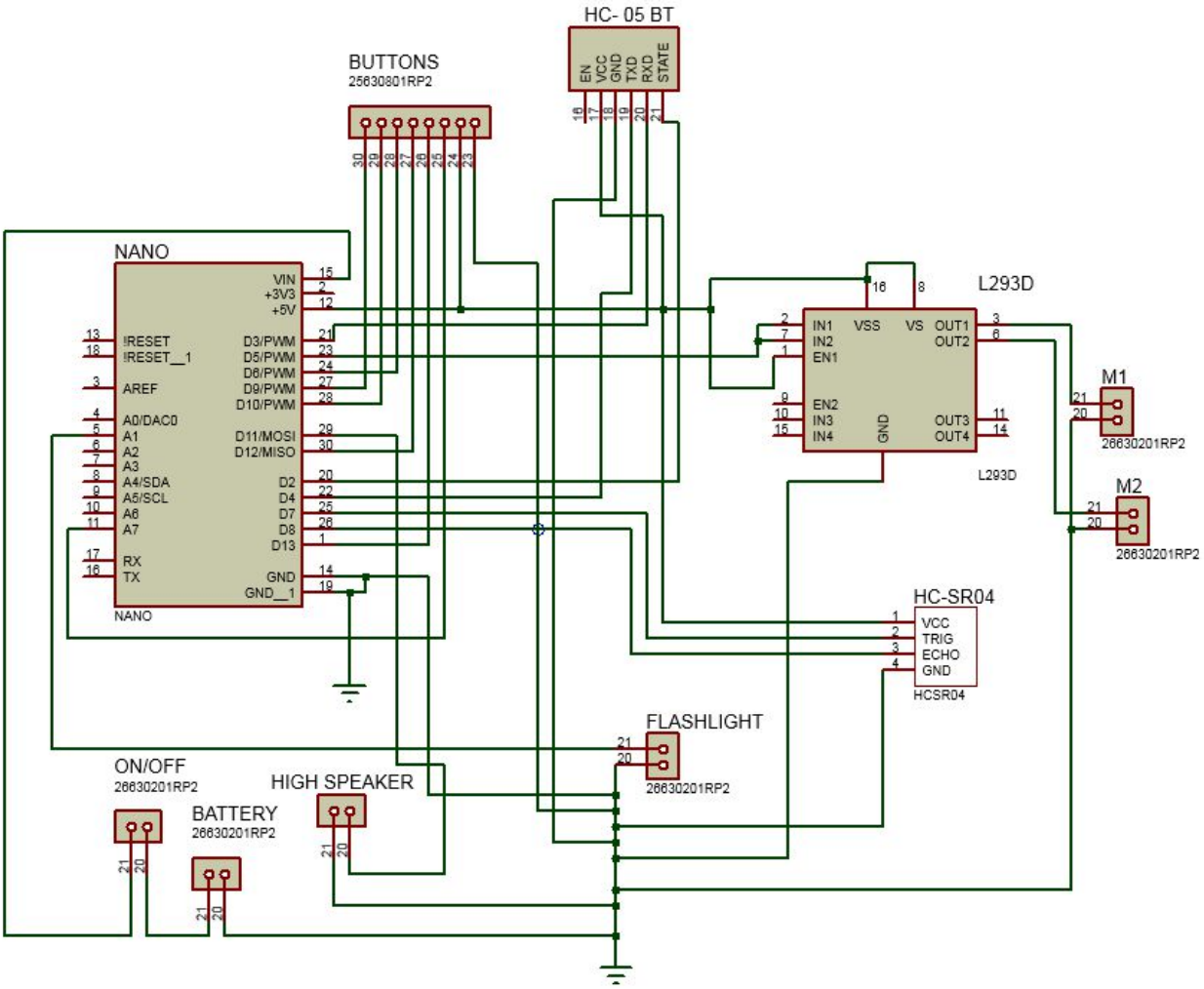


FIGURE 2.42 – Le circuit final Arduino en Proteus

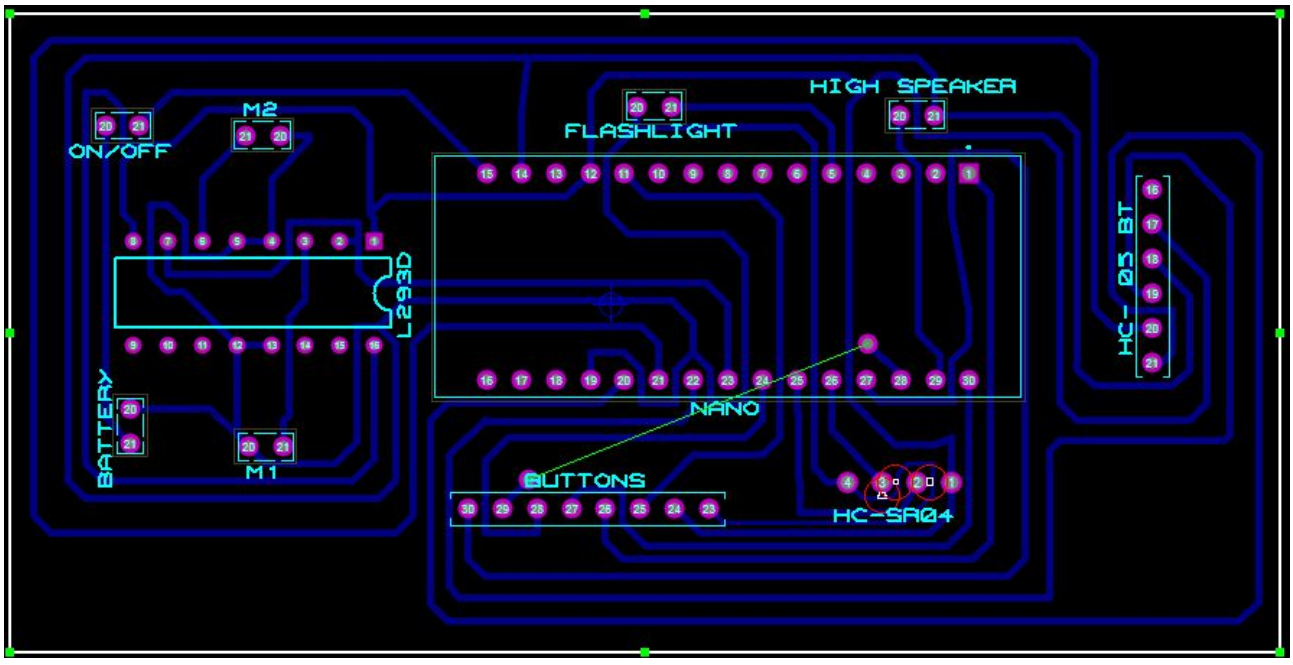


FIGURE 2.43 – Layout du PCB final Arduino en Proteus

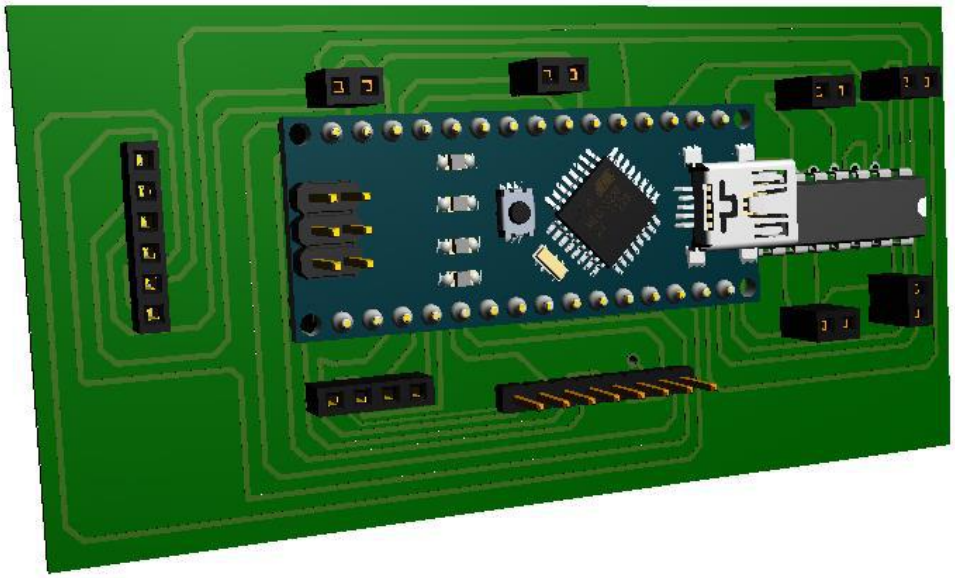


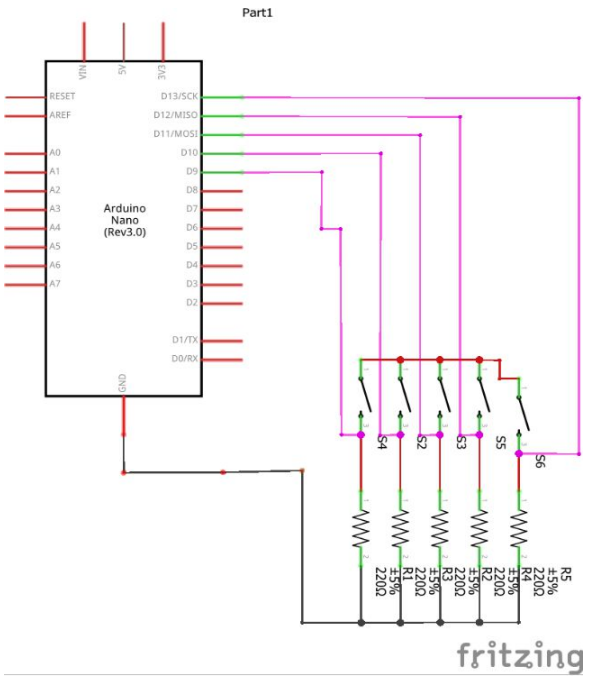
FIGURE 2.44 – Modèle 3D du PCB final d'Arduino

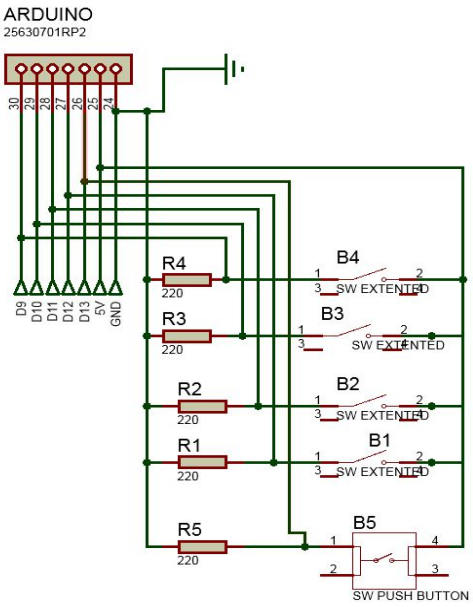
2.7.2.4 Circuit des boutons

Quand nous avons commencé le processus de conception des PCB nous n'avions aucune idée du nombre de PCB qu'il y aura, nous voulions faire seulement un seul PCB qui va contenir tous les composants mais en conséquence il serait sorti grand en taille et qui va affecter considérablement la taille de la canne, et aussi la canne sera actionnée par l'utilisateur via des boutons de sorte qu'un PCB les contenant doit être proche de la surface supérieure de la canne, et en utilisant seulement un seul PCB n'aurait aucun sens , nous avons décidé alors de faire deux PCBs
Nous allons maintenant parler du processus de fabrication !

Tout comme le circuit Arduino il y avait beaucoup de changements le long de la fabrication de ce circuit, le premier schéma à suivre était celui dessus

Nous avons utilisé 5 boutons pour exécuter différentes tâches pour faire fonctionner la canne, mais dans le schéma, nous pouvons remarquer que les boutons sont liés à l'arduino et nous avons mentionné plus tôt que l'arduino et les boutons sont séparé, C'est pourquoi nous avons connecté les boutons à un support qui sera connecté à ceux qui se trouve dans le circuit Arduino

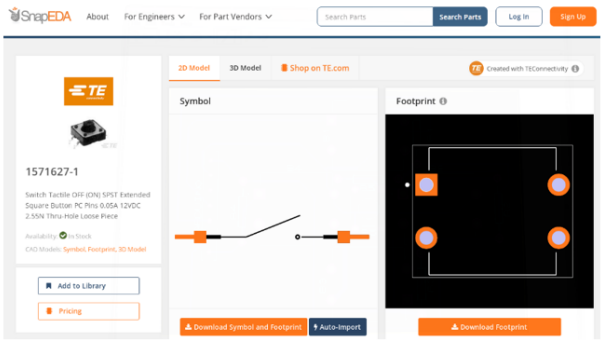


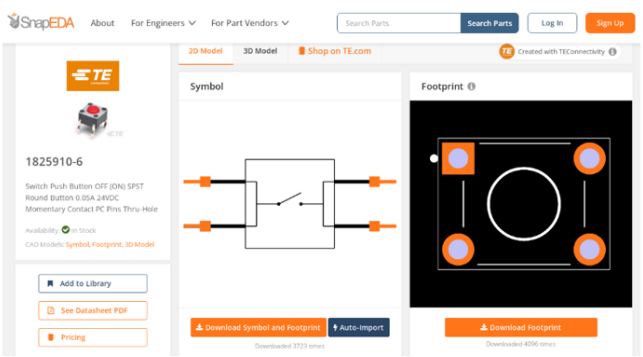


Quand nous avons voulu choisir les packages de chaque bouton nous allons utiliser dans le mode PCB layout nous n'avons pas pu trouver ceux appropriés dans la bibliothèque fournie qui va correspondre à ceux que nous allons utiliser et puisque les modèles de visualisation 3d vont être exportés et utilisés comme une base de conception 3d ils sont dû être précis

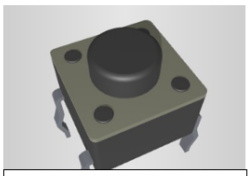
Nous avons utilisé des boutons-poussoirs momentanés : les 4 sur les côtés étaient gros et celui au milieu était petit afin que nous puissions économiser l'espace le maximum possible

Nous sommes allés à snapEDA pour télécharger l'empreinte et le modèle 3d des boutons et les utiliser dans le pcb layout

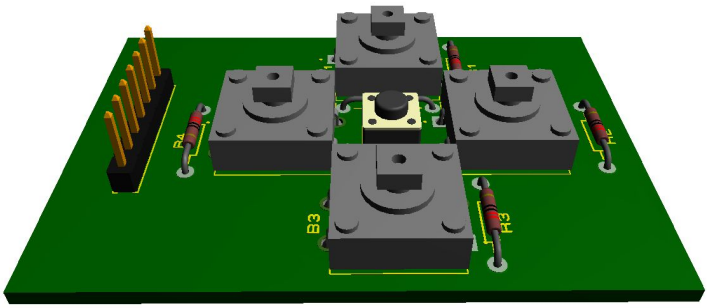




Bouton pour les cotés

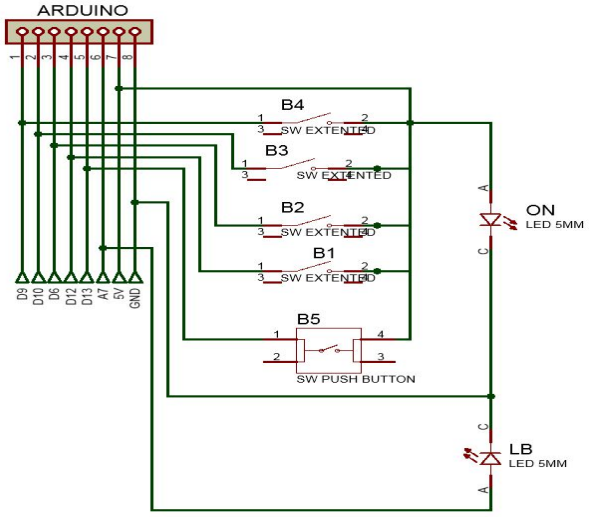


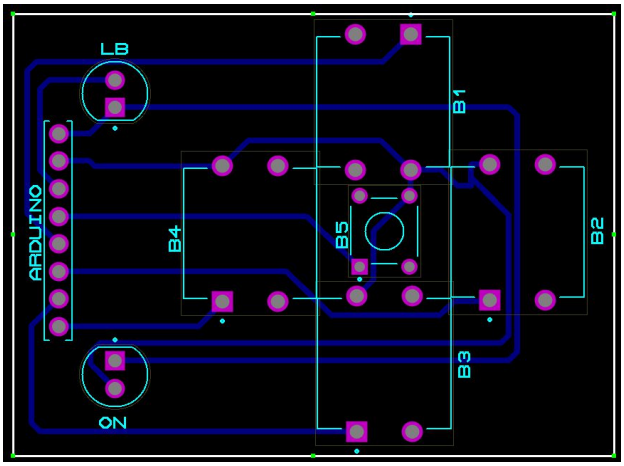
Bouton milieu

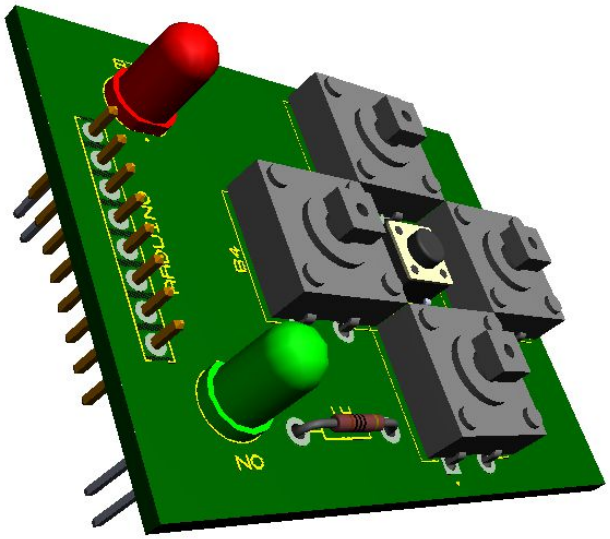


La première version du circuit imprimé ressemblait à ceci, mais nous avons éventuellement apporté quelques changements

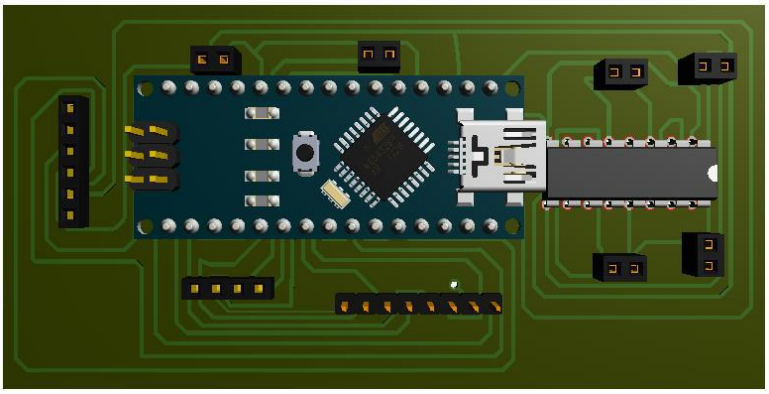
Deux leds ont été ajoutées : une pour indiquer batterie faible, et l'autre pour indiquer que la canne est sous tension Résistance supprimée : nous allons utiliser les résistances intégrées sur l'Arduino (la raison était mentionnée dans le chapitre précédent)

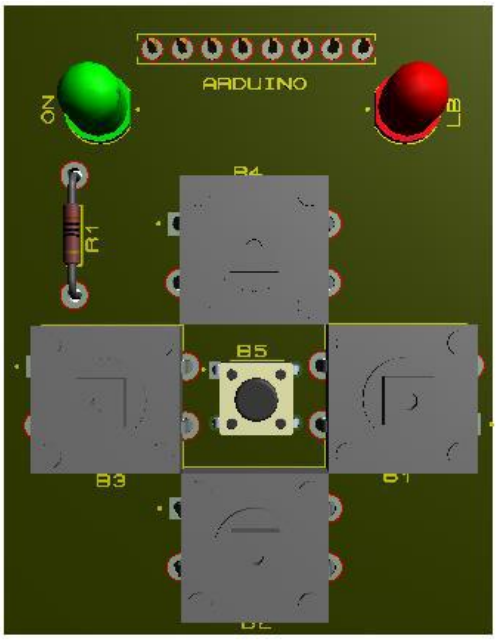






Après avoir fini la conception des circuits imprimés nous avons fini avec ces deux modèles 3d





Nous les utilisons en cours de fabrication de la canne dans le logiciel de conception 3D choisi !

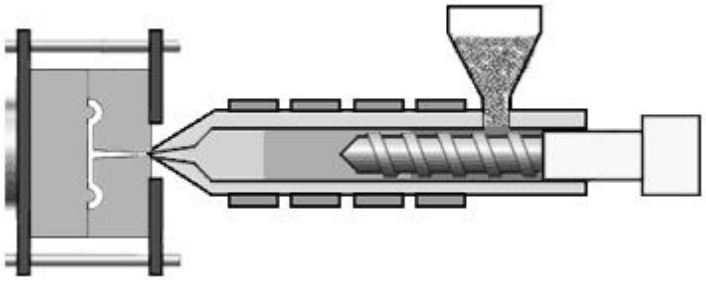
2.7.3 Conception 3D de bras

Nous avons dû donner à nos pcbs une vie en leur donnant un boîtier afin que nous puissions assurer la portabilité et la convivialité du produit final : la canne intelligente.

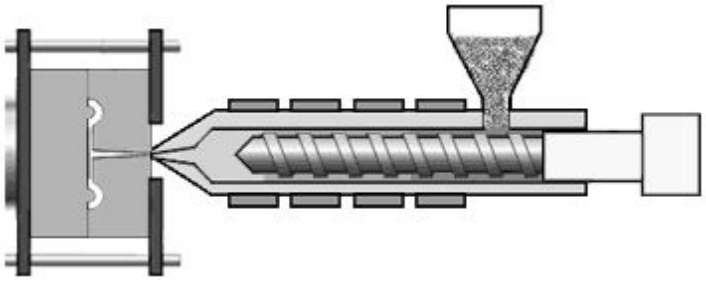
D'abord nous avons pensé au polystyrène comme notre matière de base que nous allons faire notre canne, c'est un matériau léger, facilement accessible pour nous et aussi il nous a donné la capacité.

Pour le façonner comme notre souhait, mais nous n'étions pas entièrement satisfaits de ce matériau : il aurait l'air bon marché et amateur, nous voulions quelque chose de mieux, quelque chose qui correspondrait à la norme de l'industrie de la fabrication de produits de consommation à main : le plastique est le bon matériau.

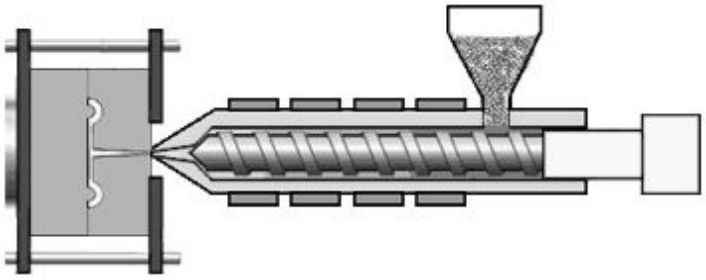
Il existe beaucoup de techniques pour fabriquer des objets à partir de plastique, mais en tant qu'étudiants, nous ne pouvons pas aller avec les techniques complexes industrielles qui nécessitent des années d'expertises comme par exemple le moulage par injection : Le procédé de moulage par injection consiste à chauffer et à injecter du plastique sous pression dans un outil de moulage métallique fermé, ce qui permet de fabriquer rapidement et à moindre coût des milliers, voire des millions de pièces identiques le processus est expliqué étape par étape ci-dessous pour montrer la complexité de cette technique



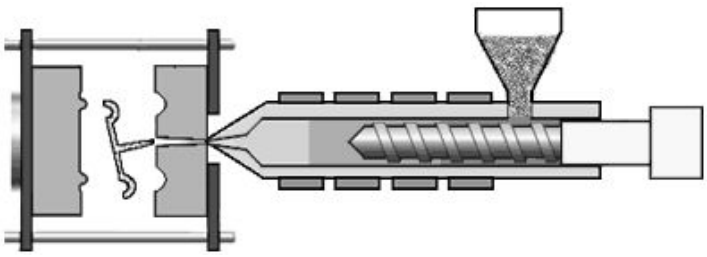
Etape 1 Les granulés de la trémie alimentent le barillet chauffé et la vis rotative.
Le matériau fondu par la chaleur, le frottement et la force de cisaillement est forcé par un clapet



Etape 2 Après avoir été déplacée vers l'arrière par le plan de matière à l'avant, la vis est poussée vers l'avant par un bélier hydraulique.



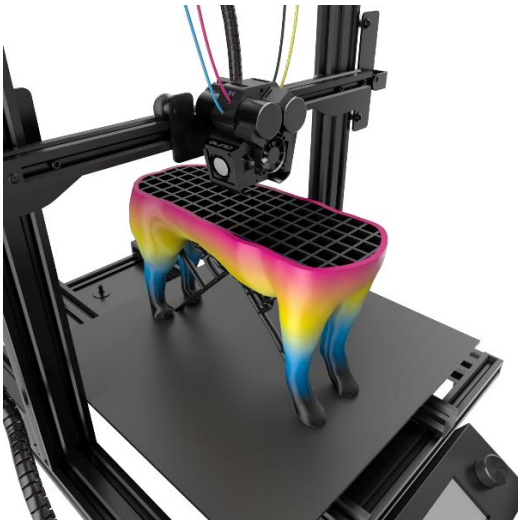
Etape 3 L'outil est maintenu fermé sous pression jusqu'à ce que le matériau plastique refroidisse et durcisse dans la cavité de l'outil de moule.
C'est souvent la partie la plus longue du processus de moulage par injection



Etape 4 La vis commence à reculer pour le prochain moulage. L'outil s'ouvre alors et la pièce plastique finie est éjectée.
L'outil est fermé et le processus de moulage par injection recommence à 1.

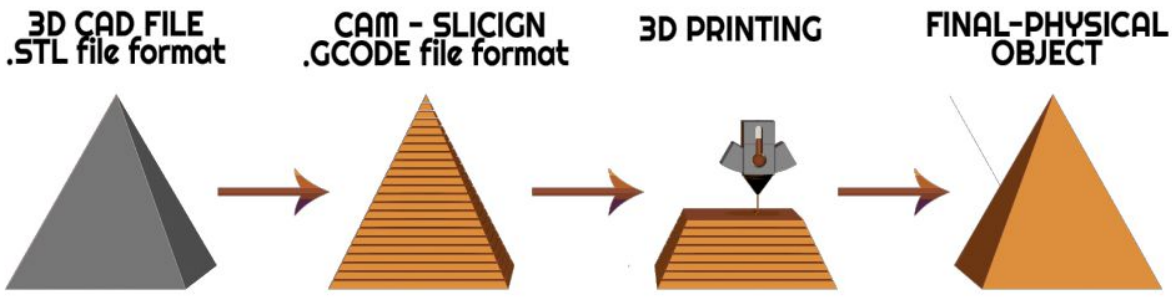
Ce processus nécessite des machines spéciales de fabrication et en tant qu'étudiants, nous ne pouvons pas suivre cette procédure, c'est pourquoi nous avons opté pour l'impression 3D :

2.7.3.1 Impression 3D



L'impression 3D ou la fabrication additive est un processus de fabrication d'objets solides tridimensionnels à partir d'un fichier numérique.
La création d'un objet imprimé en 3D est réalisée à l'aide de processus additifs. Dans un processus additif, un objet est créé en posant des couches successives de matière jusqu'à ce que l'objet soit créé. Chacune de ces couches peut être vue comme une coupe transversale finement découpée de l'objet.

Le processus d'impression 3D commence par un modèle 3D qui est réalisé via un logiciel 3D habituellement appelé un logiciel CAD* donc nous avons besoin d'un modèle 3D pour notre canne.



Mais le problème est qu'en tant qu'équipe de deux membres aucun de nous ne sait comment faire de 3d design, nous aurions pu engager un designer pour faire le travail pour nous, mais cela n'aurait pas de sens puisque c'est notre projet de dernière année et toutes les tâches doivent être faites par nos propres mains et cerveaux et c'est pourquoi nous nous sommes mis au défi et avons appris la conception 3d en moins d'un mois !

Pour créer un modèle 3d, nous avons commencé avec des croquis 2d, nous avons dessiné comment nous avons imaginé la canne pour ressembler à un morceau de papier et tout comme les pcbs conception il y avait beaucoup de changements en cours de route, le premier croquis que nous avons trouvé était celui ci-dessous.

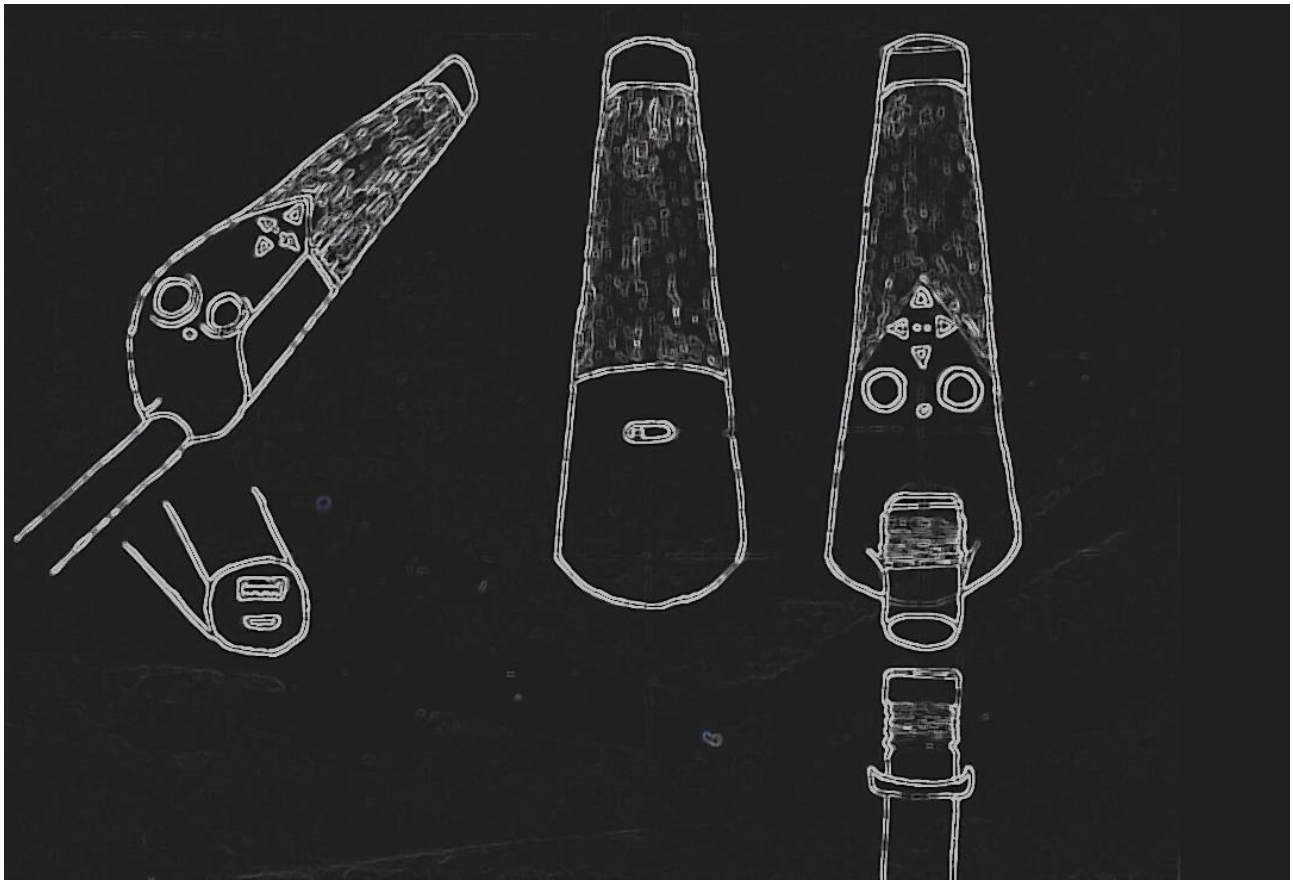


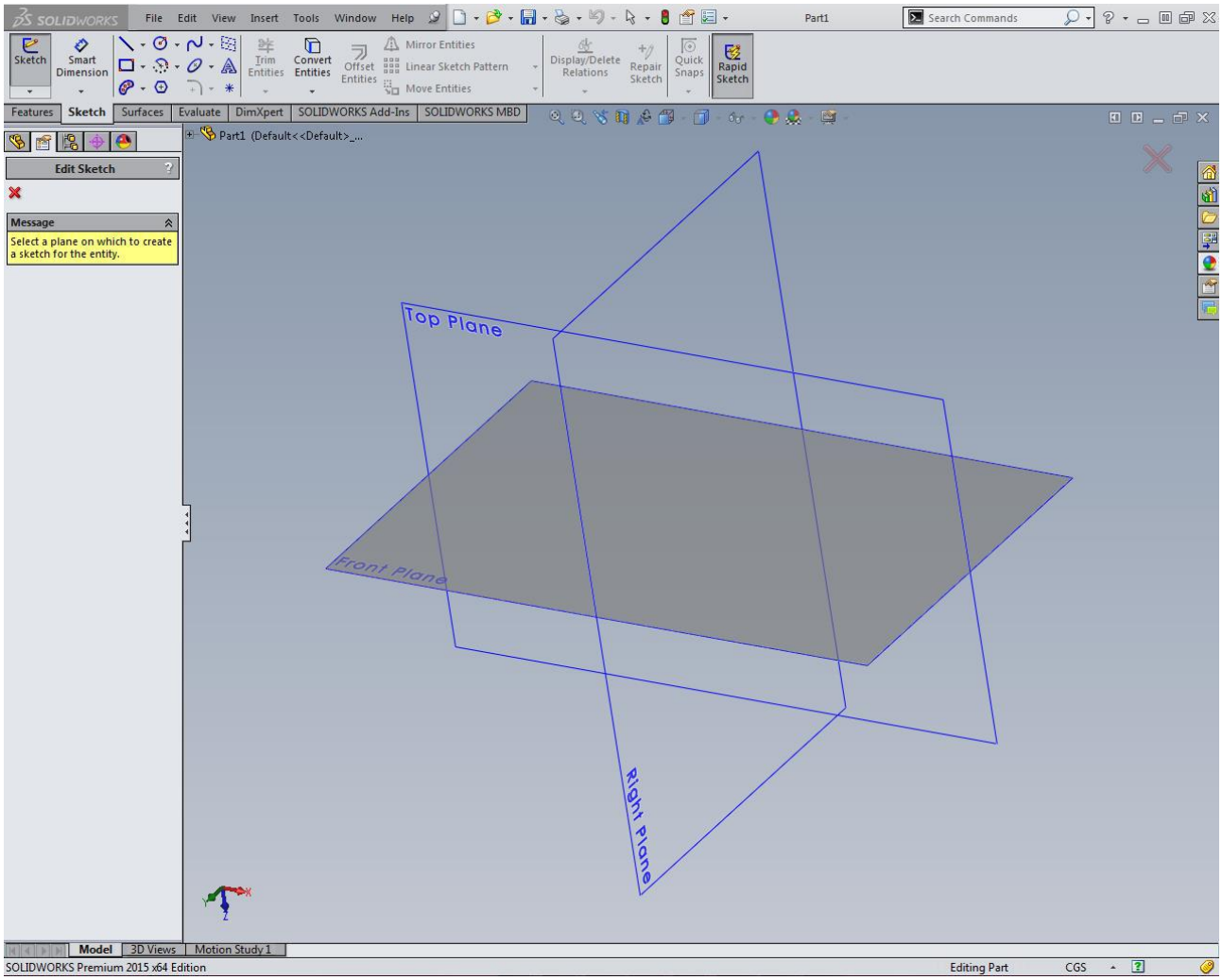
FIGURE 2.54 – Sketch 2D de la première itération de la canne

Vous pouvez voir dans ce croquis le plan supérieur des deux côtés et une vue isométrique, après le croquis 2d nous avons dû voir à quoi ça ressemblerait en 3d donc nous passons au logiciel CAD

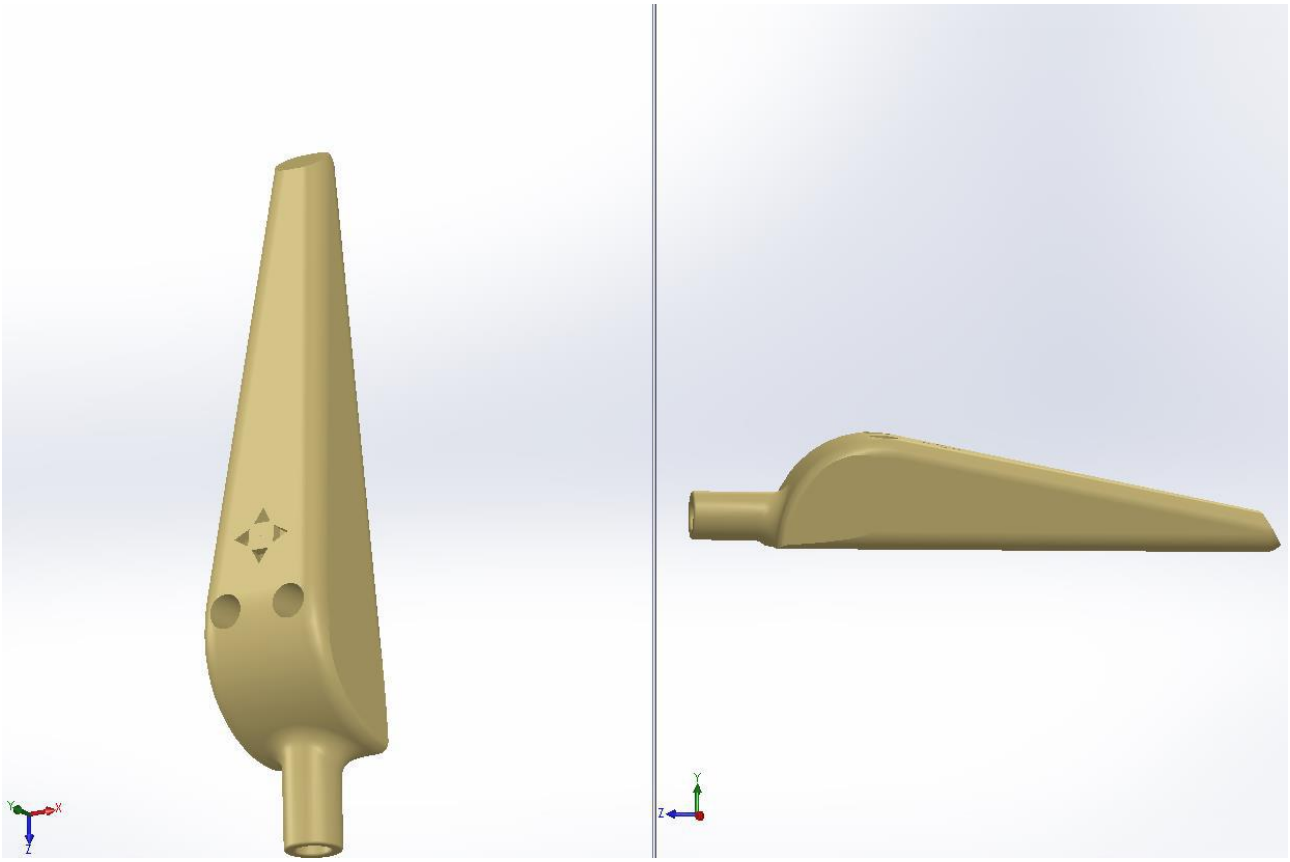
Il y avait quelques critères que nous avons fait que le logiciel devait avoir afin que nous puissions travailler avec lui :

1. Le logiciel devait être interopérable avec notre logiciel de conception de cartes Proteus, de sorte que les modèles 3D exportés puissent y être importés et prêts à fonctionner.
2. Il devrait y avoir un soutien en ligne actif / communauté afin que nous puissions trouver des solutions à tous les problèmes / difficultés que nous allons faire face le long du chemin faisant la canne.
3. . Enfin, il faut qu'il y ait une version piratée sur Internet, car nous n'avons pas les moyens de payer le coût de propriété de l'application. La plupart des options correspondaient à notre

liste de critères, mais nous avons choisi SOLIDWORKS car nous avons trouvé l'interface utilisateur facile à comprendre et pas compliquée.

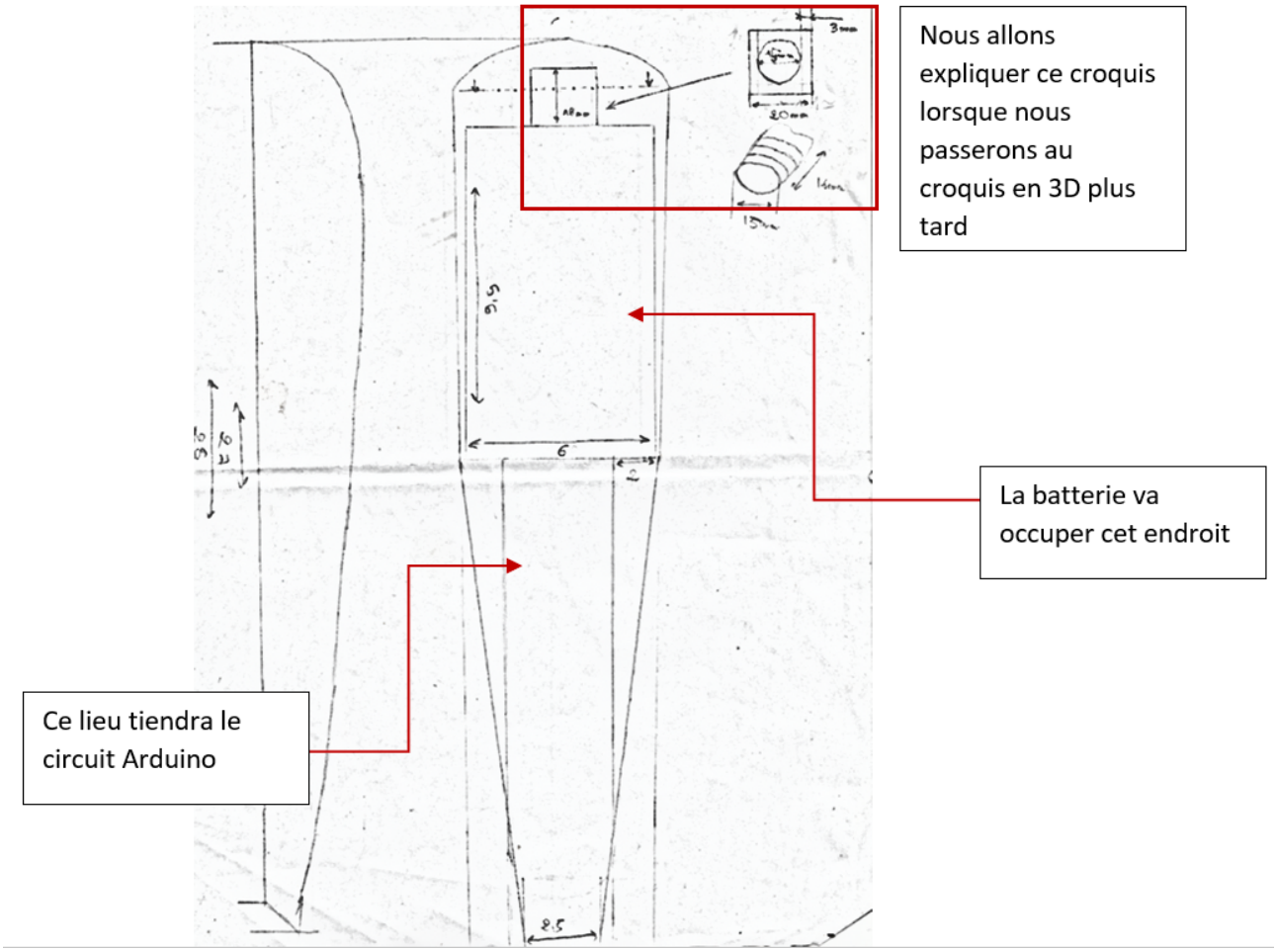


Après avoir choisi un logiciel CAD nous avons fait le premier modèle 3d de la canne juste pour donner à notre croquis 2d un peu de vie.

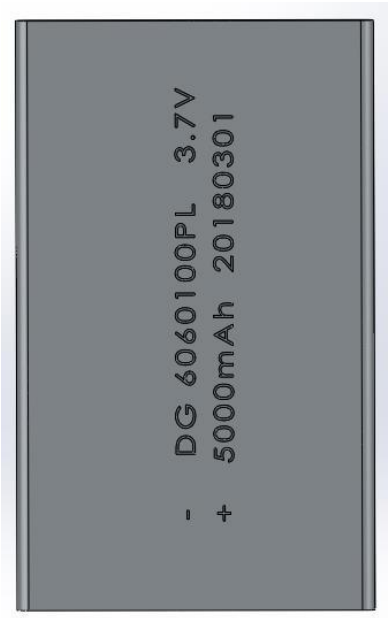


Le bouton du milieu n'a pas été inclus sur le 2d ou le croquis 3D car il a été fait avant que nous ayons ajouté le bouton dans notre circuit.

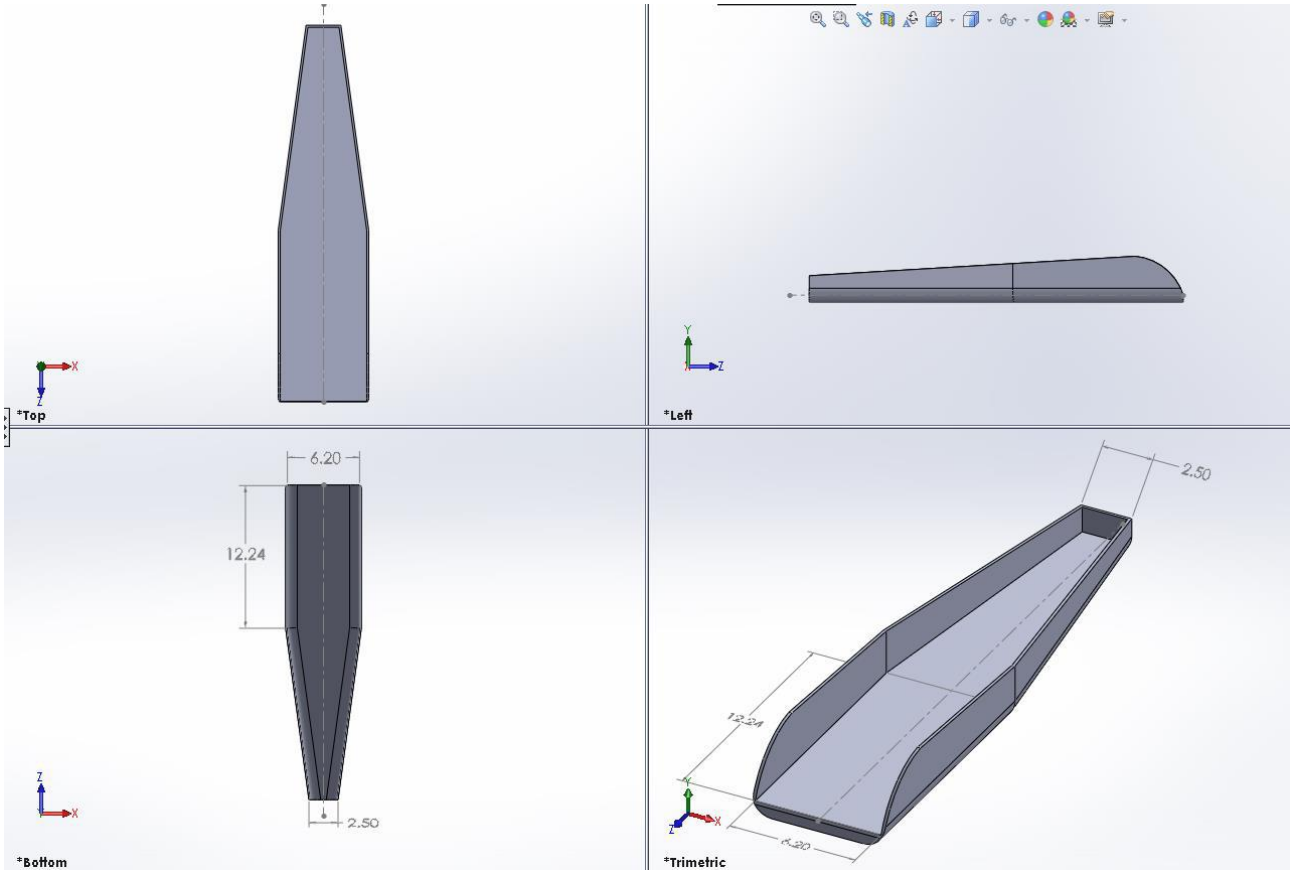
Maintenant, nous avons commencé à travailler avec des mesures réelles basées sur les principaux composants utilisés dans la canne : la batterie lipo et les deux circuits Nous avons dessiné le croquis en 2d



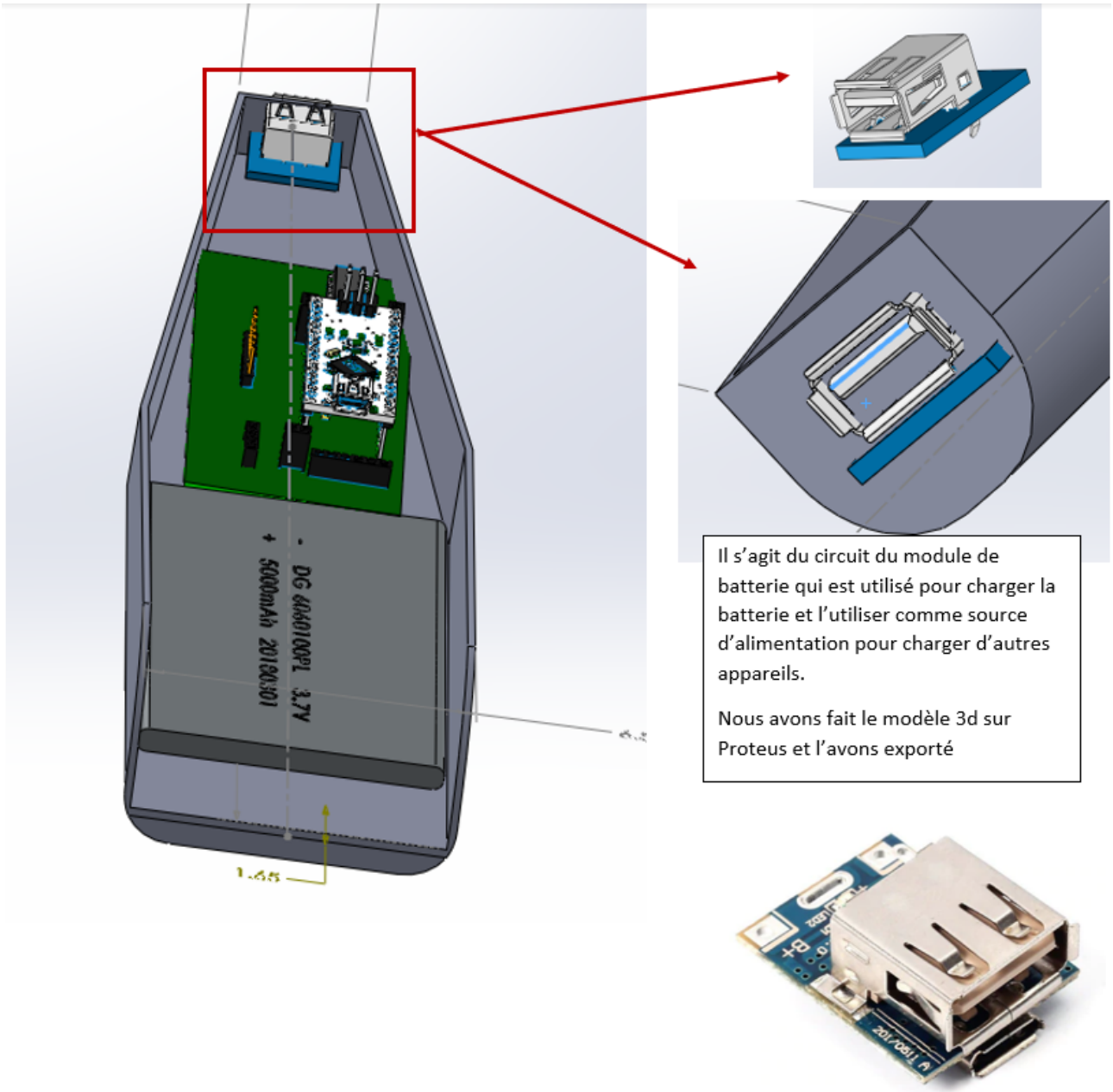
Puis nous l'avons réalisé en 3d : nous avons d'abord dû concevoir la batterie que nous allons utiliser parce qu'elle sera le point de départ de notre conception (le circuit Arduino et les boutons aussi, mais le modèle 3d est déjà fait en Proteus)



après, nous concevons la partie inférieure



Nous montons ensuite la batterie et les modèles arduino sur la partie inférieure



Il s'agit du circuit du module de batterie qui est utilisé pour charger la batterie et l'utiliser comme source d'alimentation pour charger d'autres appareils.

Nous avons fait le modèle 3d sur Proteus et l'avons exporté

Les boutons pcb doivent être accessibles à la main de l'utilisateur à elle doivent être situés dans la partie supérieure de la canne

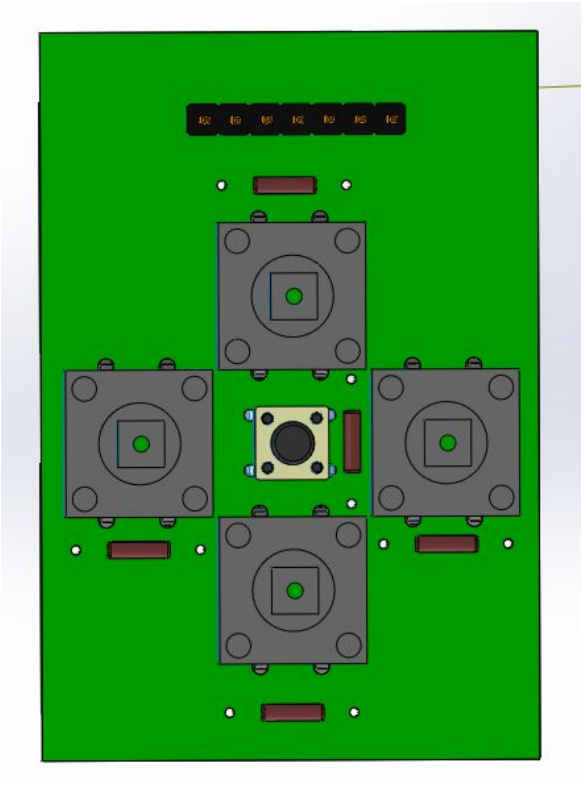


FIGURE 2.57 – Première version du modèle 3d du circuit imprimé des boutons dans SolidWorks

Nous avons donc fait une partie supérieure qui tiendrait ce circuit imprimé et en même temps être sur la partie inférieure pour former une forme solide ensemble

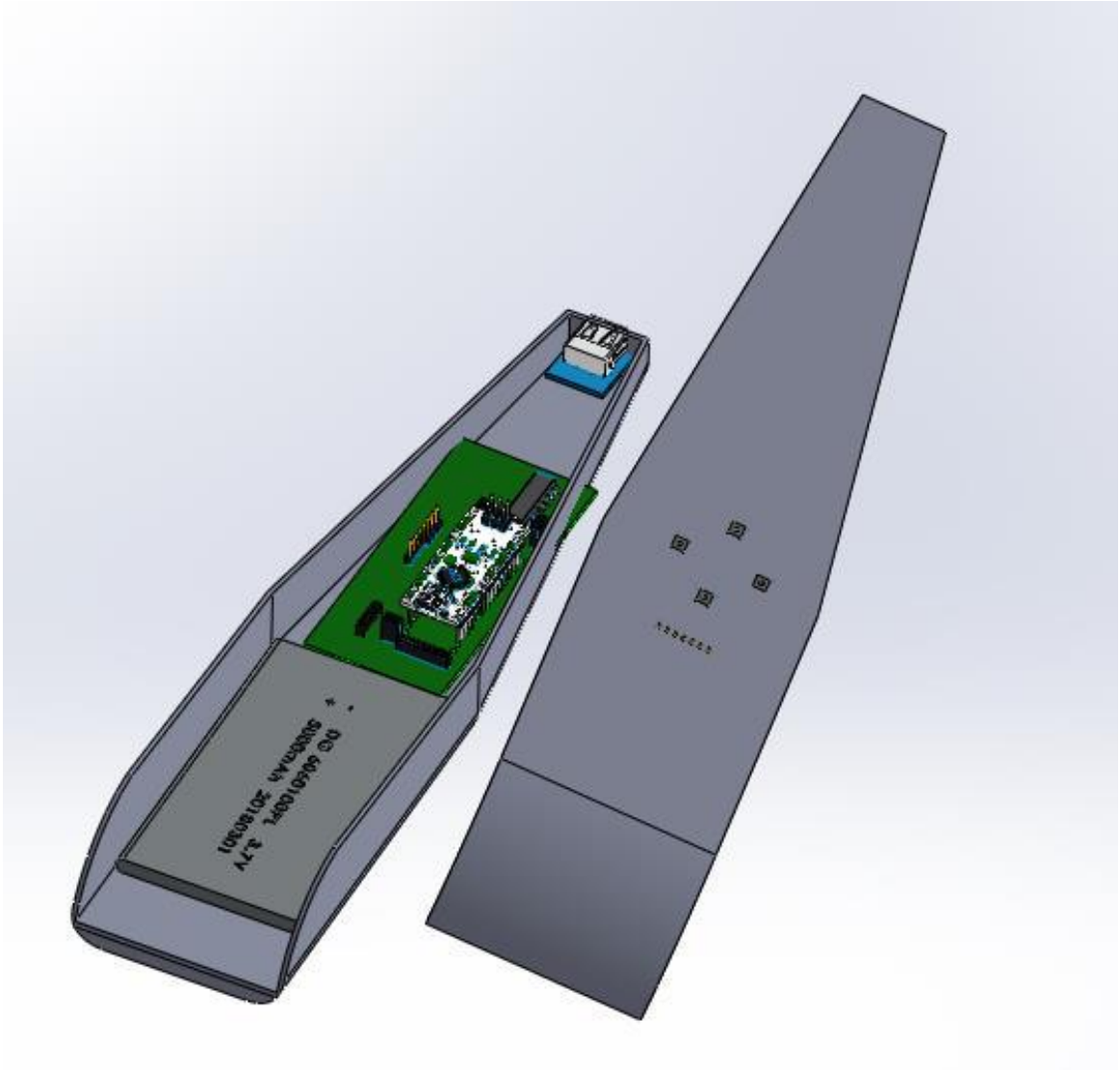
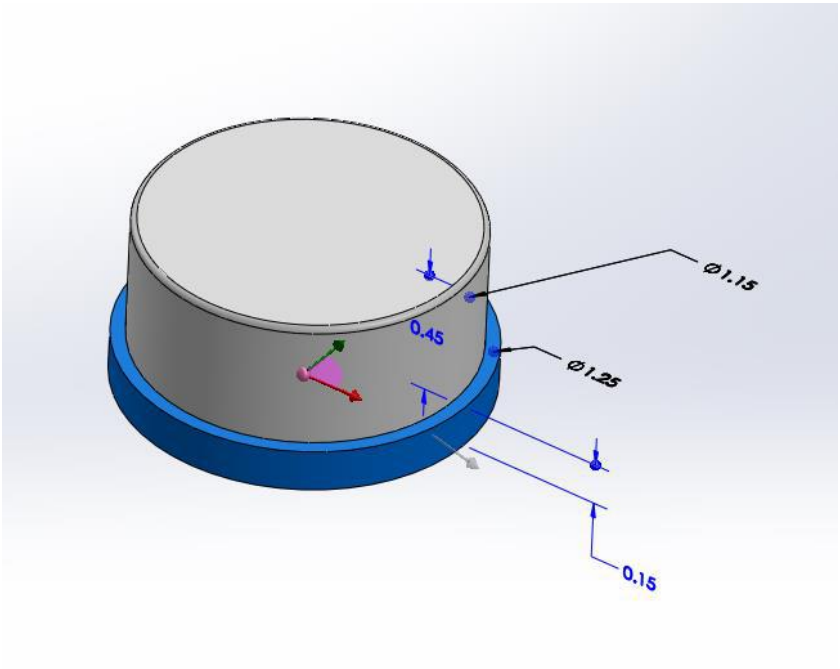


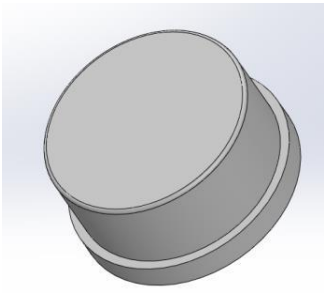
FIGURE 2.58 – Première version du modèle 3D de la bras

Les boutons que nous avons utilisés (ceux sur les côtés) étaient inclus avec des bouchons (figure a) ,nous avons dû concevoir les mêmes bouchons afin pour pouvoir les inclure dans le modèle 3d de circuit des boutons

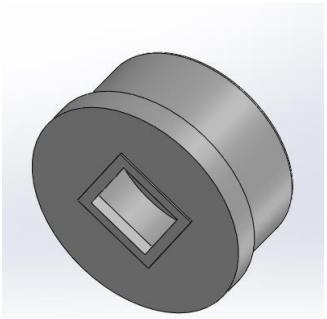
(a)



(a) Les mesures sont exactement comme celles utilisées dans la vie réelle.



(b)



(c)

Après avoir fini les bouchons, nous les avons montés sur le dessus de la carte comme ils sont dans la vraie vie

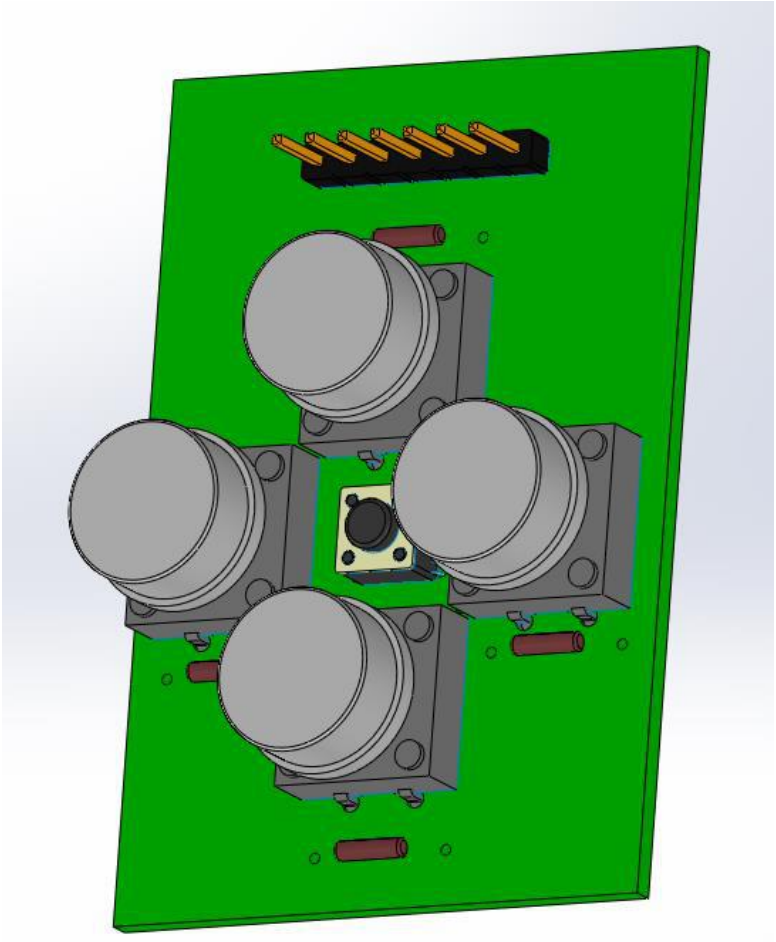


FIGURE 2.61 – Circuit boutons 3d avec les bouchons

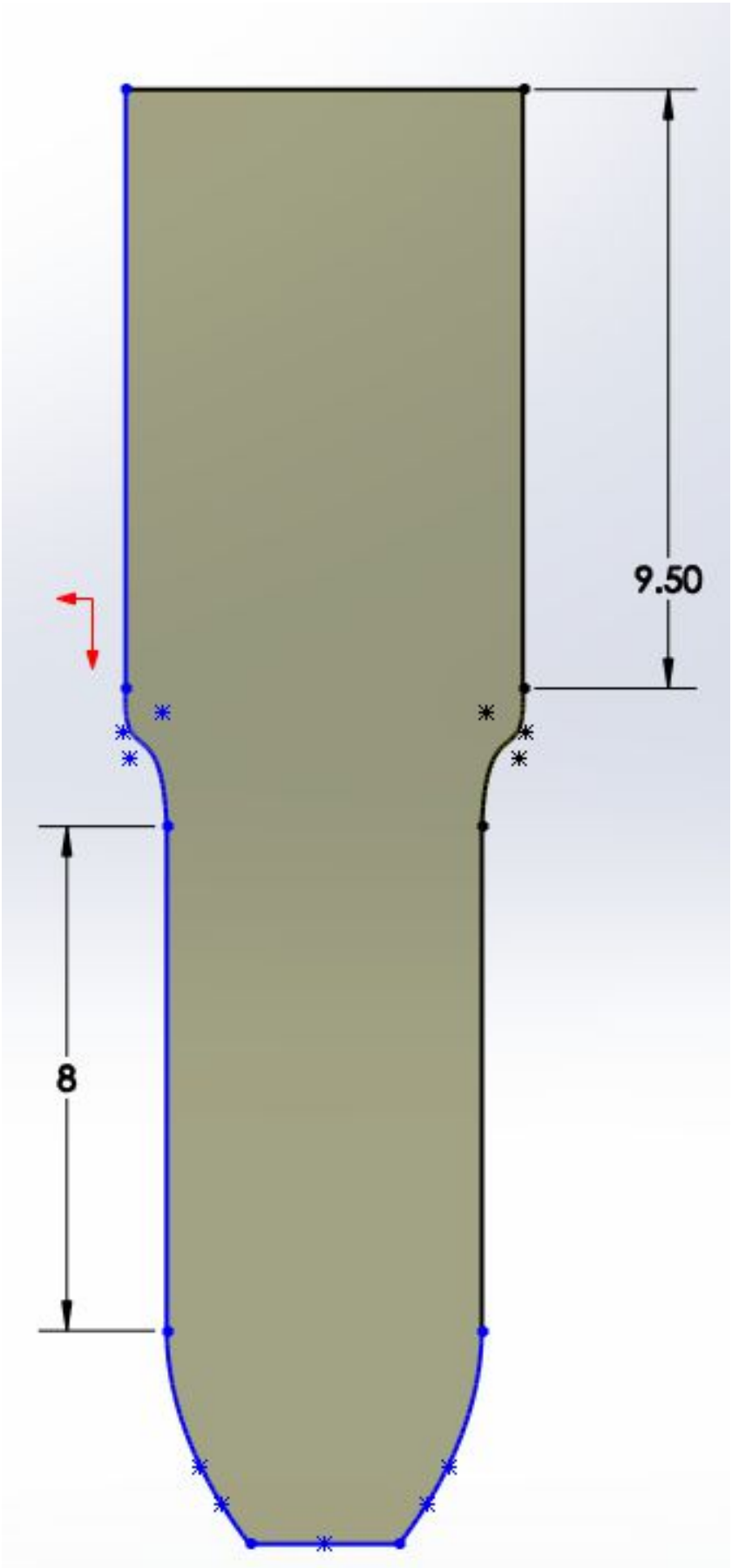
FIGURE 2.62 – Première itération du bras

Puis on a tout monté et on a fini avec ça

Notre premier design 3D est terminé et il semble incroyable. . . !
Non, ce n'est certainement pas étonnant, nous n'avons pas aimé le résultat que nous voyions : ce n'était pas attrayant comme nous l'avions prévu, c'était lourd, ennuyeux et dans l'ensemble ne valait pas l'impression 3D.

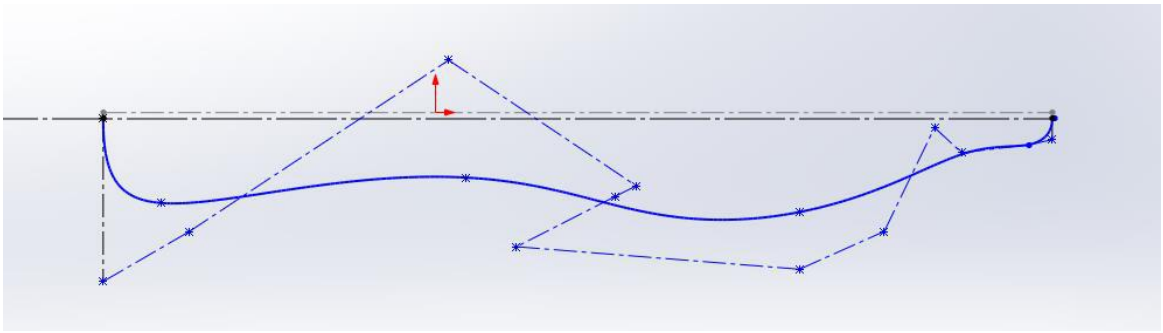
Nous avons pris une pause car nous avons mis beaucoup d'efforts dans la conception ci-dessous (il n'a peut-être pas aimé qu'il ait fallu beaucoup d'effort à faire, mais nous n'avions aucune expérience donc nous avons dû apprendre les techniques et les bases dès que possible pour réaliser que mais apparemment ce n'était pas assez)
Nous avons donc pris une semaine qui a était dédié pour apprendre des techniques plus avancées qui nous aideront à réaliser notre vision.

La base de notre conception 3d (parties supérieure et inférieure) était ce coquis ci-dessous

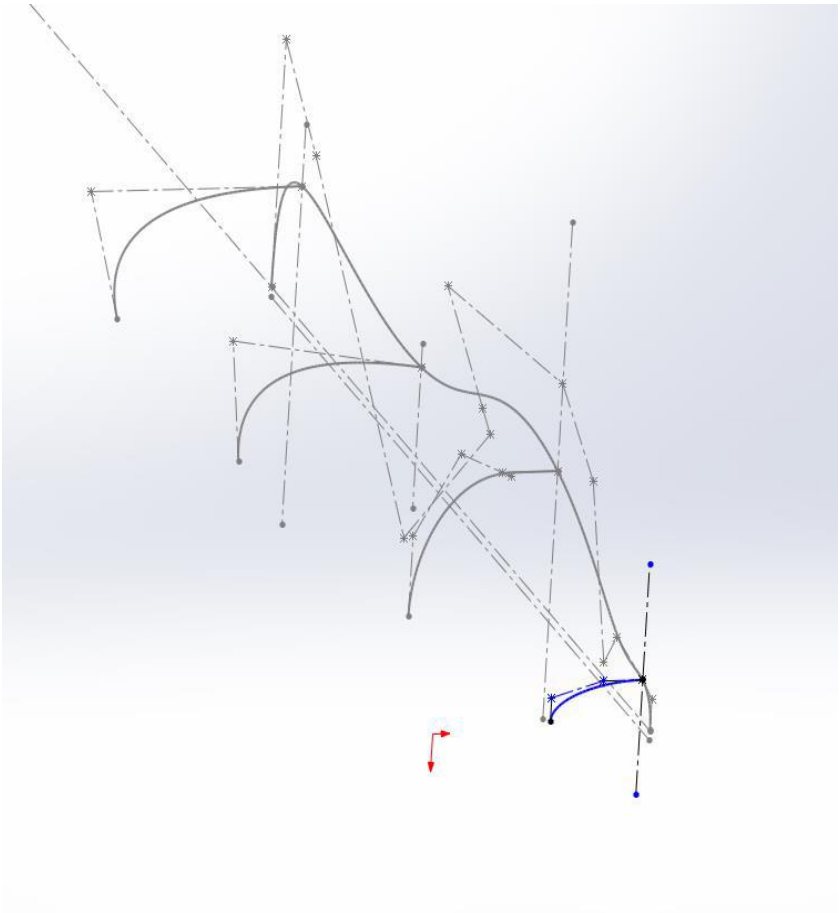


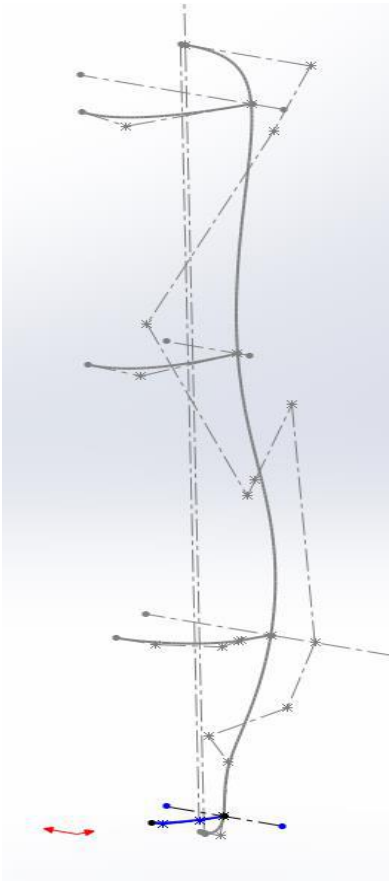
La partie supérieure a dû être commencée de ce croquis et la même chose s'applique à la partie inférieure.
Commençons par la partie inférieure car c'est celle qui va contenir la plupart des composants.
Le problème avec la première version était qu'elle ressemblait à une boîte !
Il ne ressemblait pas à quelque chose qui a été fait pour être portable, et il serait certainement laisser quelques cicatrices sur la main de tout utilisateur qui le détient pendant un certain temps.

Des courbures étaient nécessaires dans la conception pour la rendre plus ergonomique

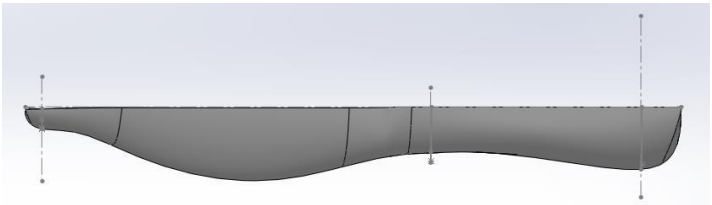


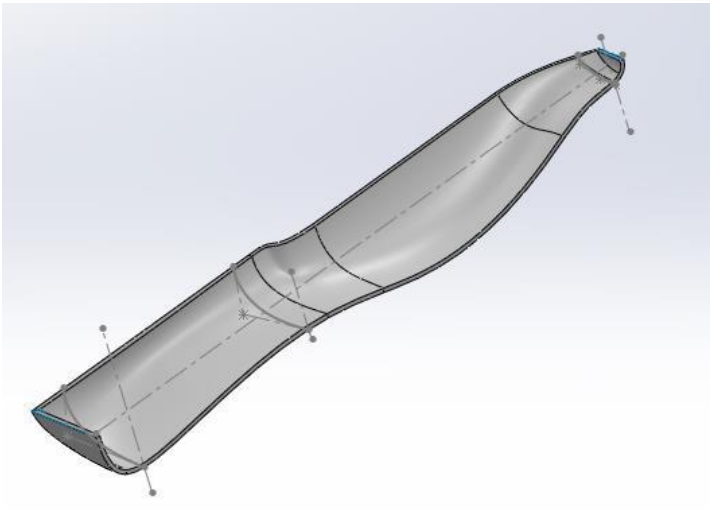
Cette courbure va être au milieu de la partie inférieure, et le long d'autres courbures sur le côté, il va donner le plus bas une meilleure forme.

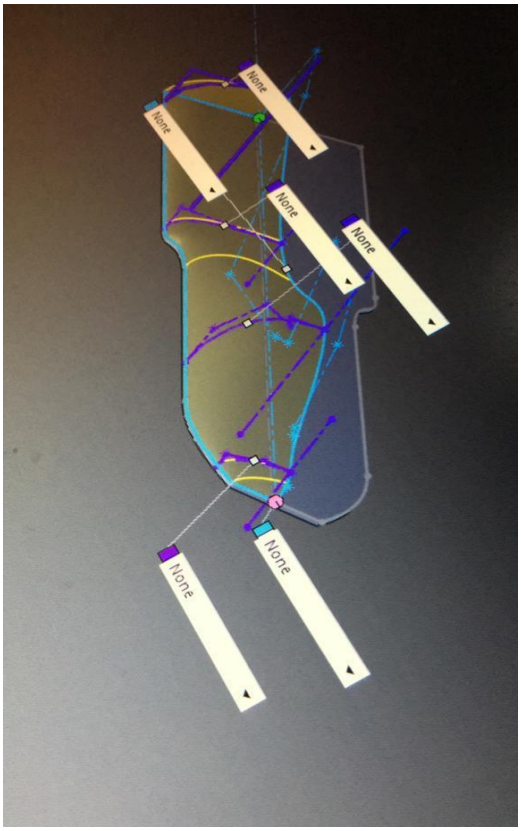


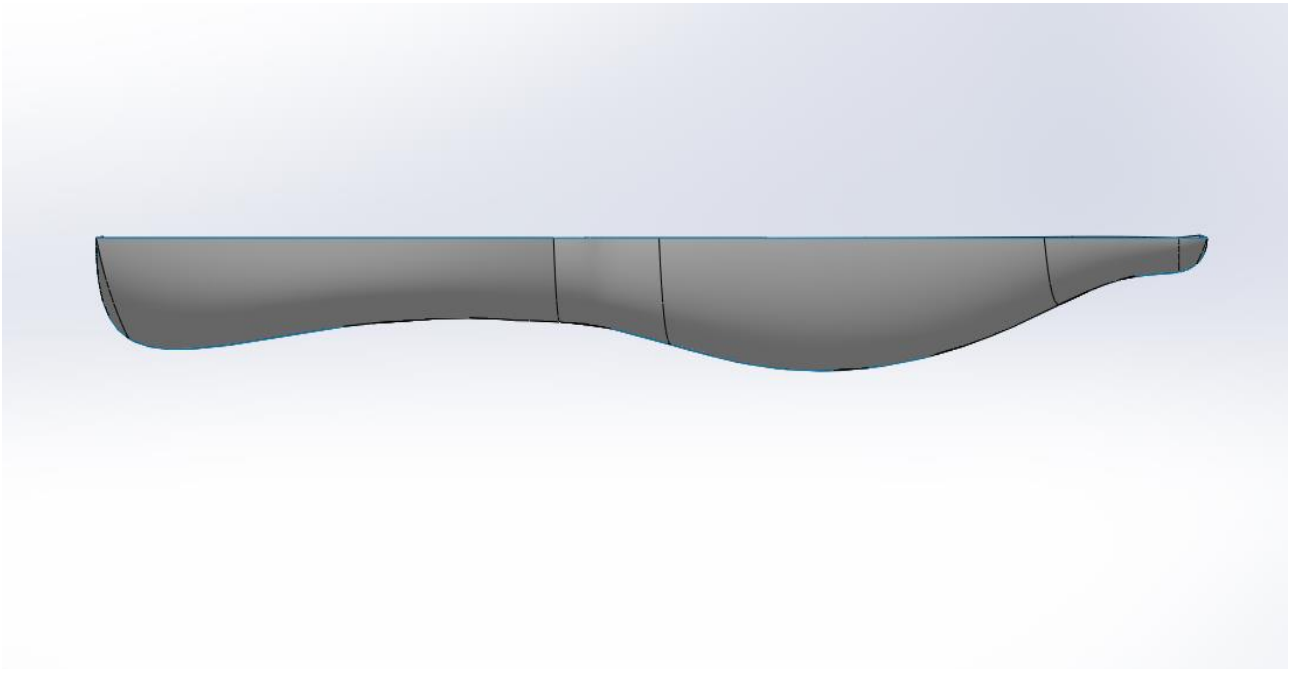


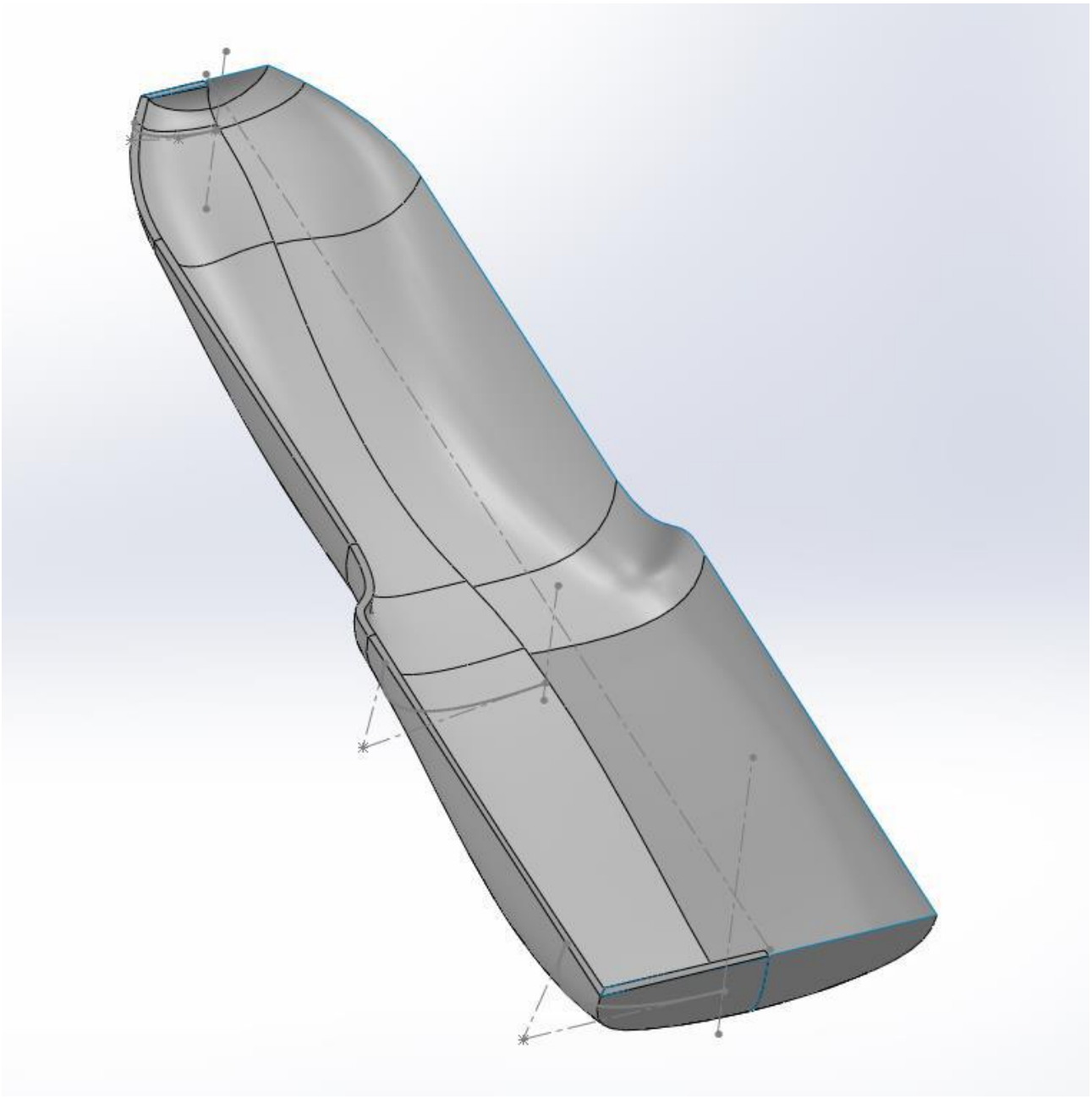
Nous ne faisons les courbes latérales que d'un côté car nous allons utiliser une fonctionnalité dans SOLIDWORKS appelée « miroir », cela signifie essentiellement que nous pouvons reproduire des croquis d'un côté à l'autre .





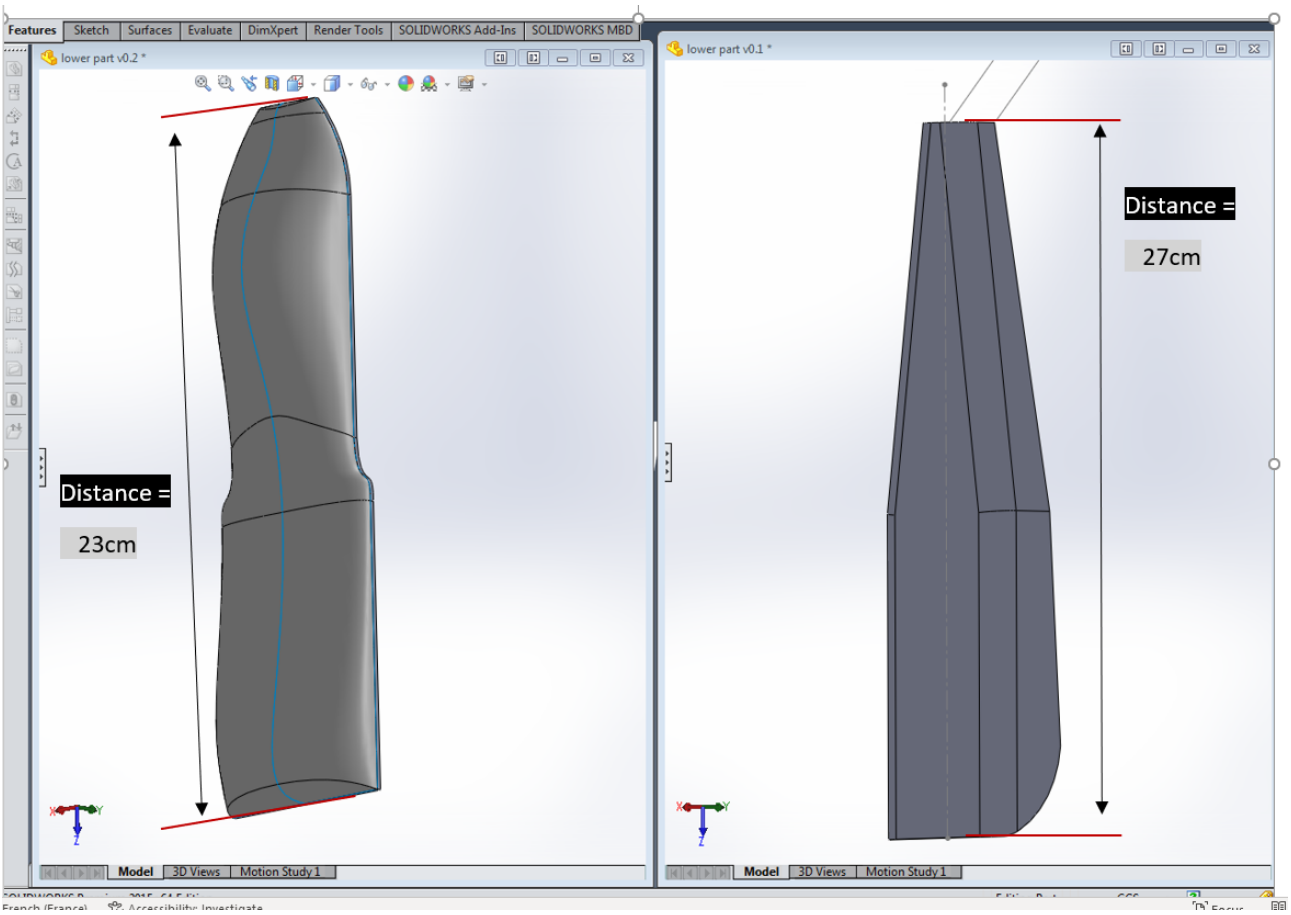

None
None
None
None
None
None





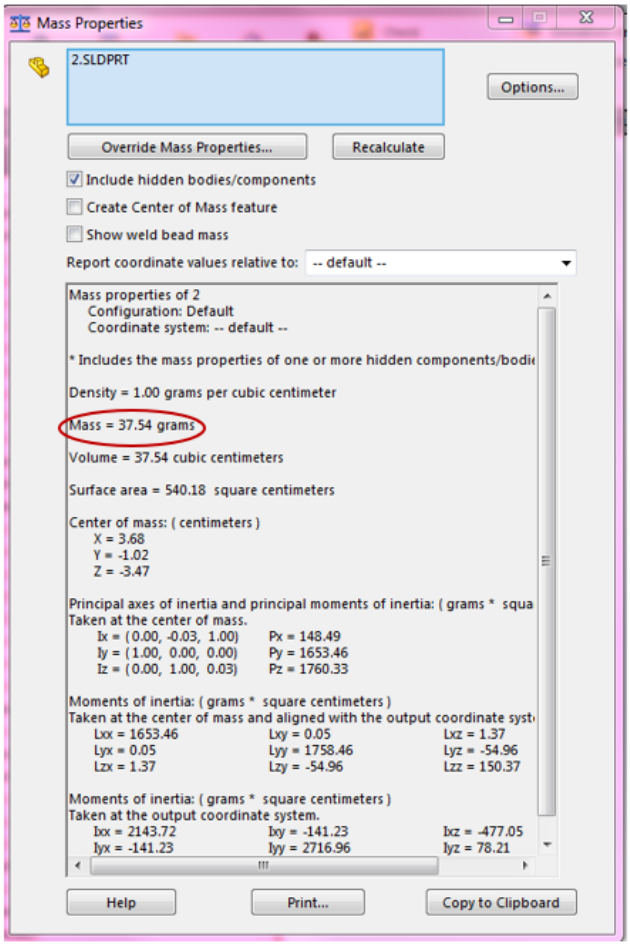
C'est ce que nous finissons avec après le miroir , la partie est creuse de l'intérieur parce qu'il doit adapter les composants

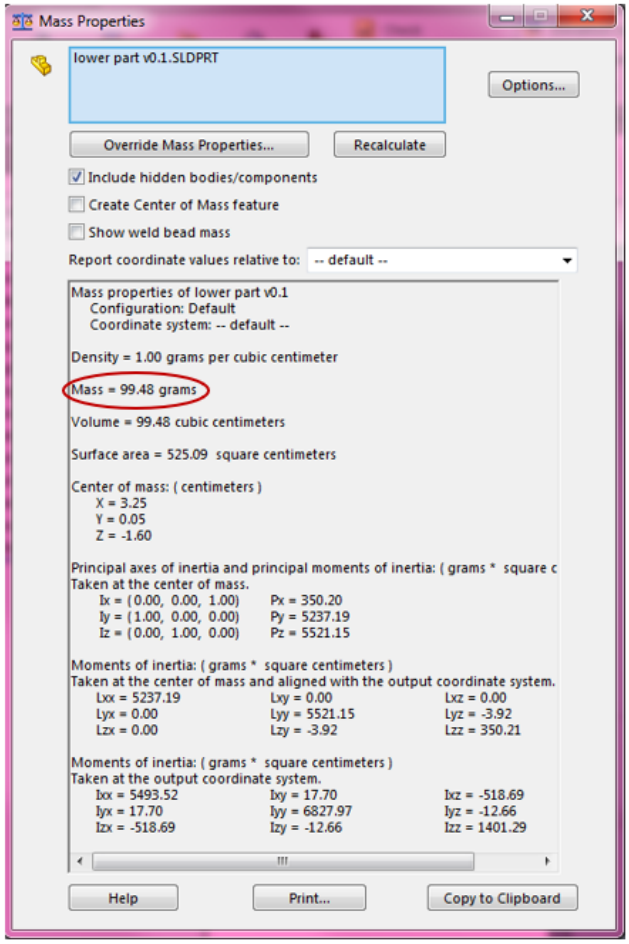
Soulignons les principales différences entre les deux versions de la partie inférieure !



Donc la différence la plus évidente est que la deuxième version a plus de courbes par rapport à la première et qui lui a donné plus le sentiment de produits de consommation (télécommande).

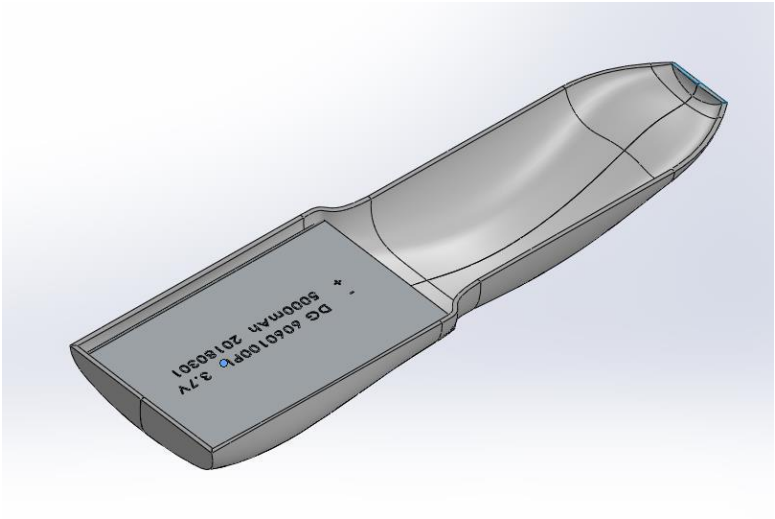
En outre, il y a une différence de hauteur de 4cm qui le rendent la partie inférieure sur tout moins lourd.



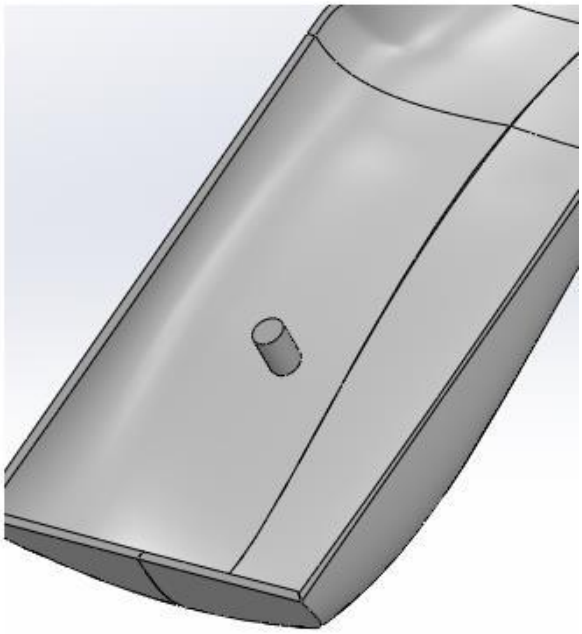


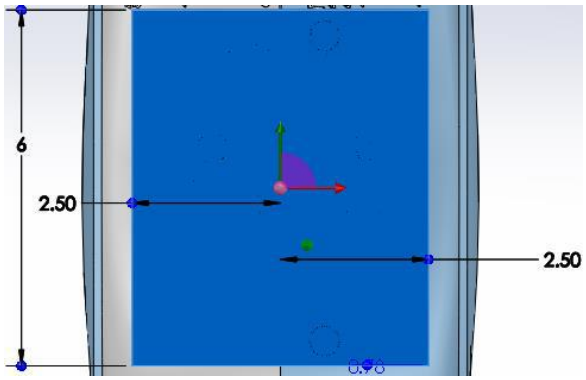
Et par moins lourd on entend une diminution de presque 70 gr entre les deux versions.

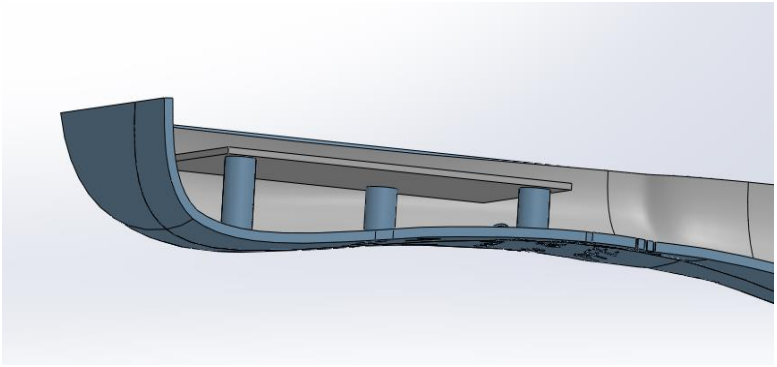
Nous devons faire des espaces pour les composants utilisés dans la canne, nous avons commencé avec la batterie lipo.



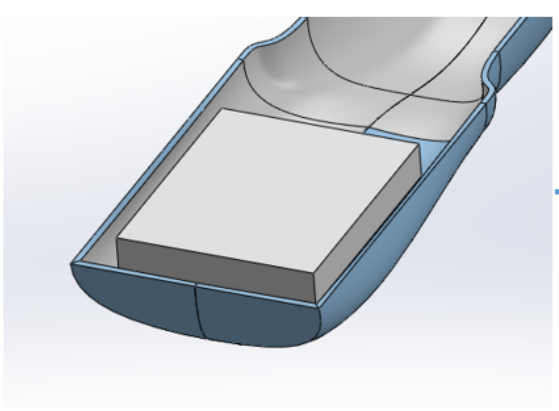
Pour tenir la batterie à sa place, nous devons faire un plateau et coller la batterie sur elle afin que nous puissions éviter tout type de mouvements indésirables

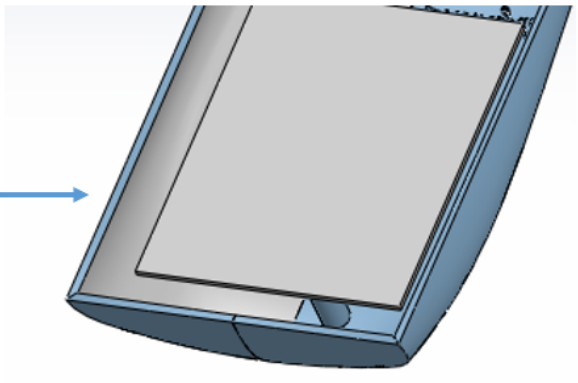


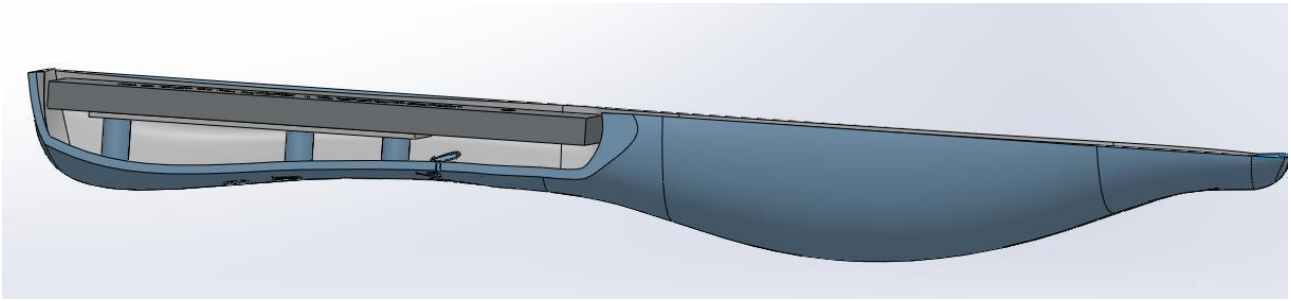


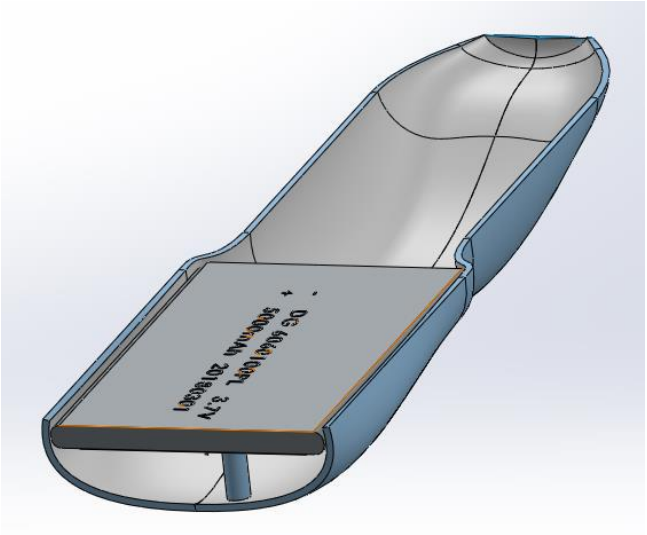


C’est le plateau dont nous avons besoin, il a 3 supports et sera préfet pour adapter la batterie sans se croiser avec la partie supérieure

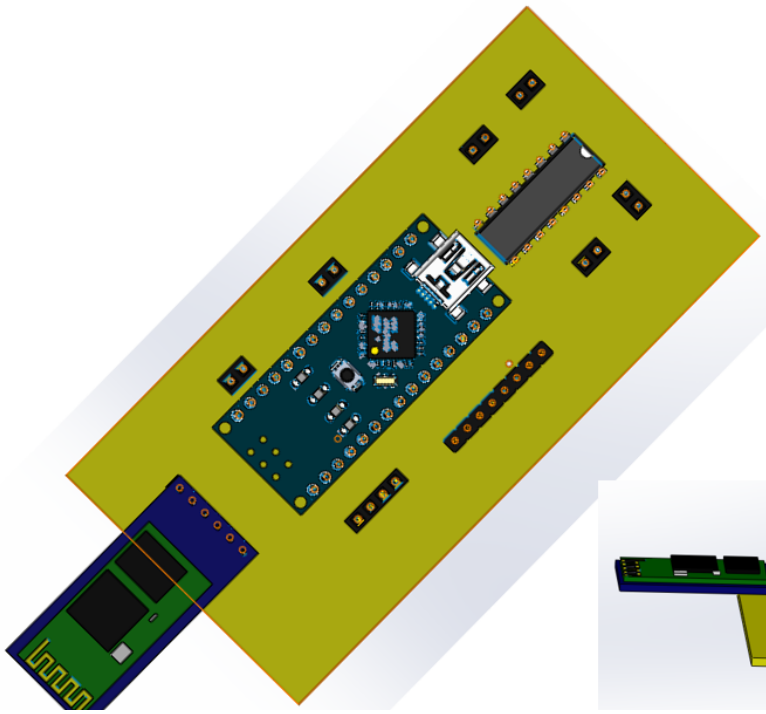




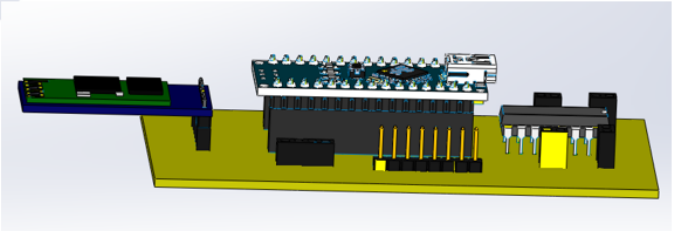




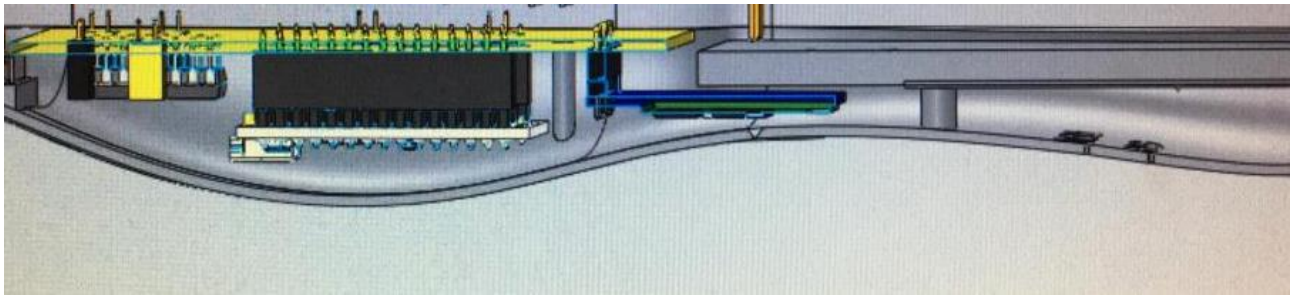
Maintenant, nous montons l'arduino pcb dans la partie inférieure.



Nous travaillons avec la version finale des circuits à ce stade, donc après avoir exporté la carte Arduino de Proteus et l'avoir importée sur SOLIDWORKS nous nous retrouvons avec ce modèle.

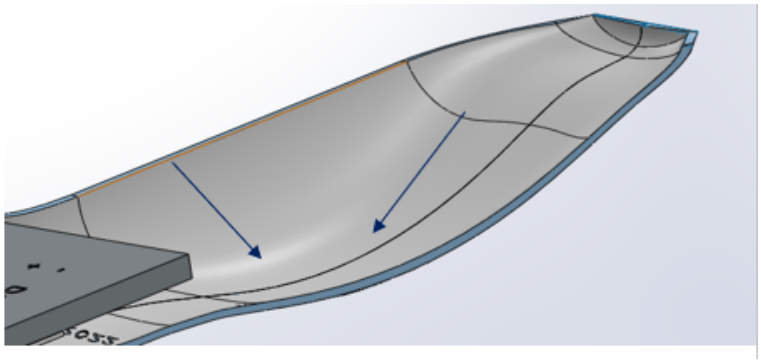


Il y avait deux possibilités de montage du circuit Arduino : il devait soit être orienté vers le haut, soit vers le bas.

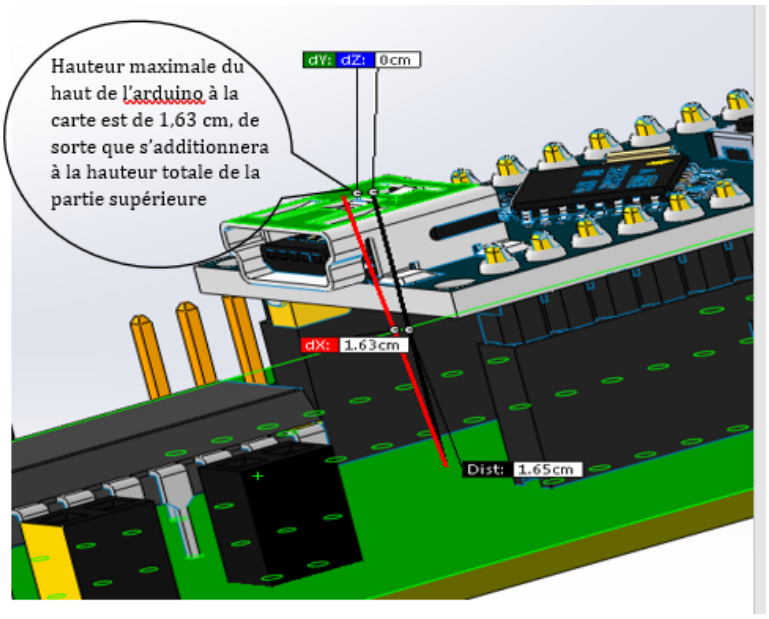


Il peut être monté comme ceci mais il y avait deux inconvénients avec cette façon :

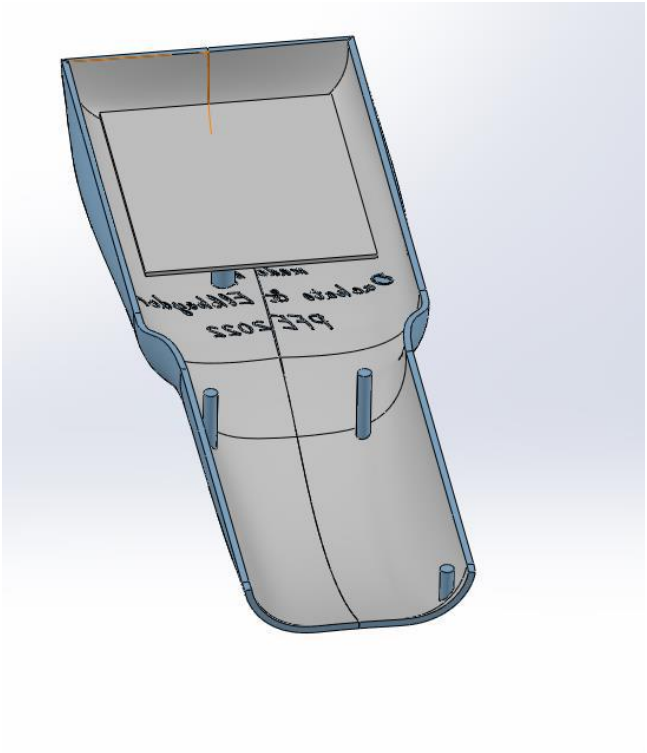
1. Nous ne profiterions pas de cette courbure qui a été faite en premier lieu pour rendre la canne confortable sur la paume de la main.

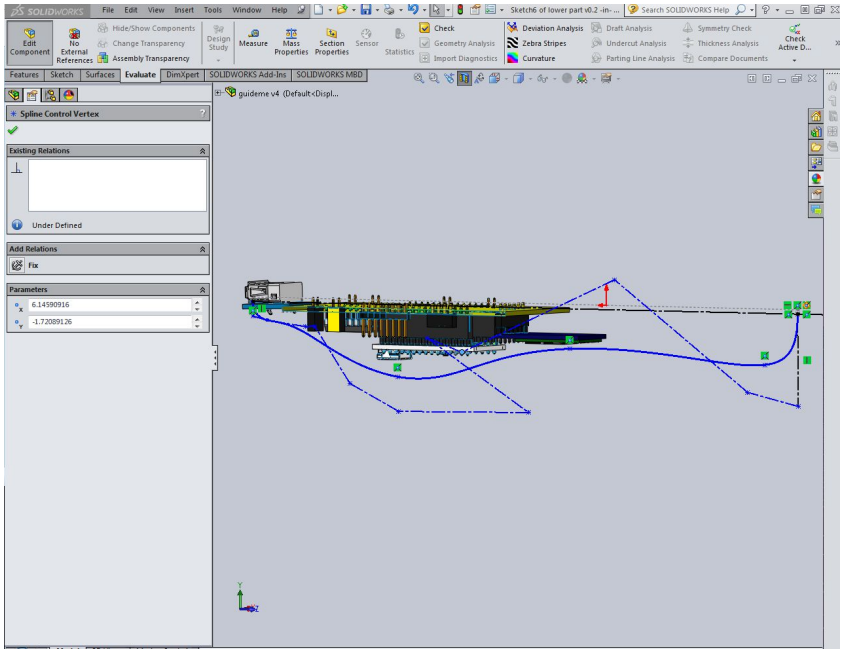


2. L'Arduino a une certaine hauteur et il affectera la hauteur de la partie supérieure.



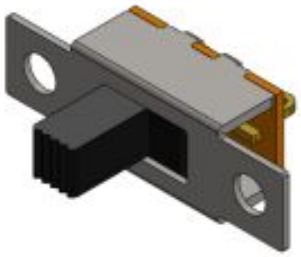
C'est pourquoi nous allons le monter face vers le bas, nous allons d'abord faire quelques trous de fixation pour visser le circuit imprimé en place !

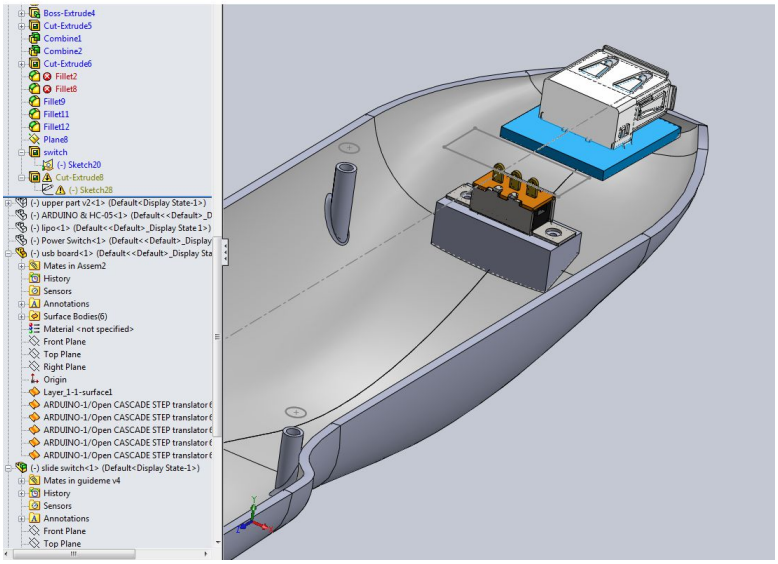


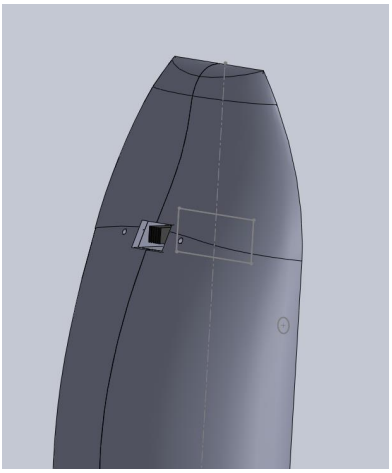


Nous avons remarqué après le montage du circuit imprimé qu'il touche la surface de la partie inférieure.
Nous avons ensuite redessiné la forme de la partie inférieure (la courbure centrale précisément) de sorte qu'il peut adapter le circuit imprimé principal sans aucune frotement

Après avoir fini la carte, nous devons maintenant monter l'interrupteur d'alimentation.
Nous allons utiliser un commutateur à glissière (figure)
Nous avons cherché un modèle 3D pour l'utiliser dans solidworks de GRABCAD
Et nous en avons trouvé un qui correspond au nôtre.

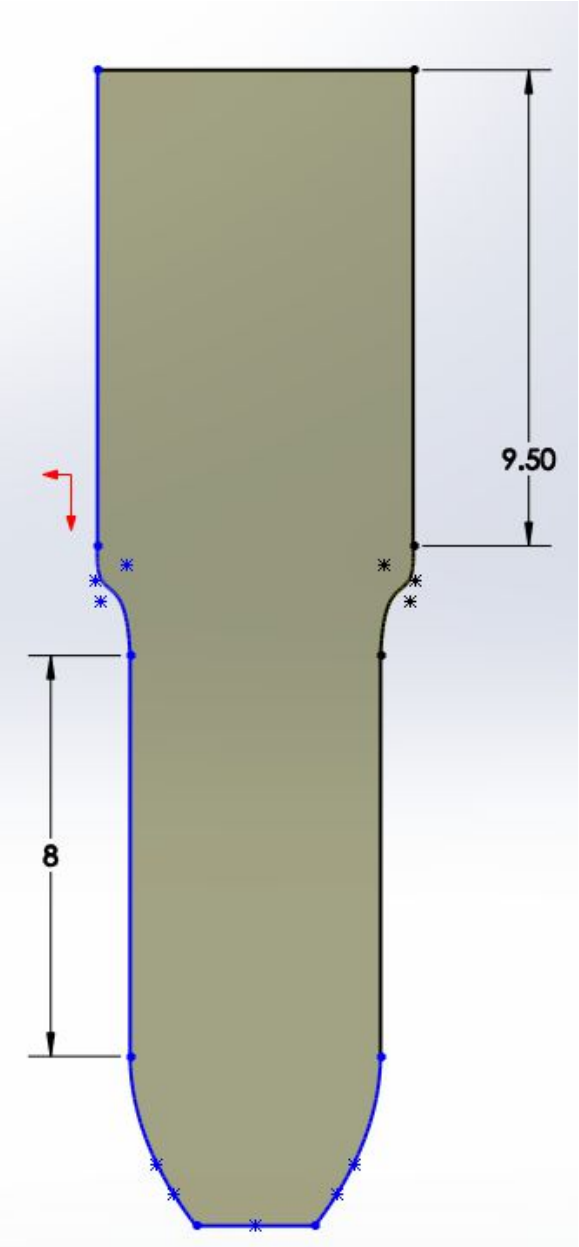


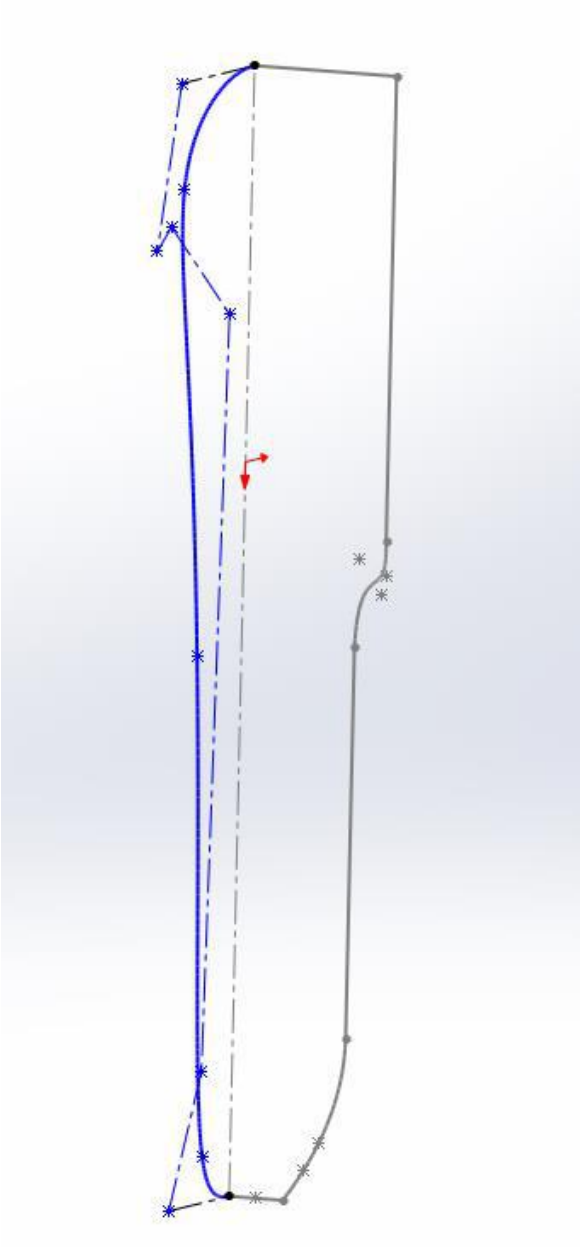




Maintenant, nous avons terminé la partie inférieure et monté toutes les pièces : batterie, circuit Arduino et le commutateur.

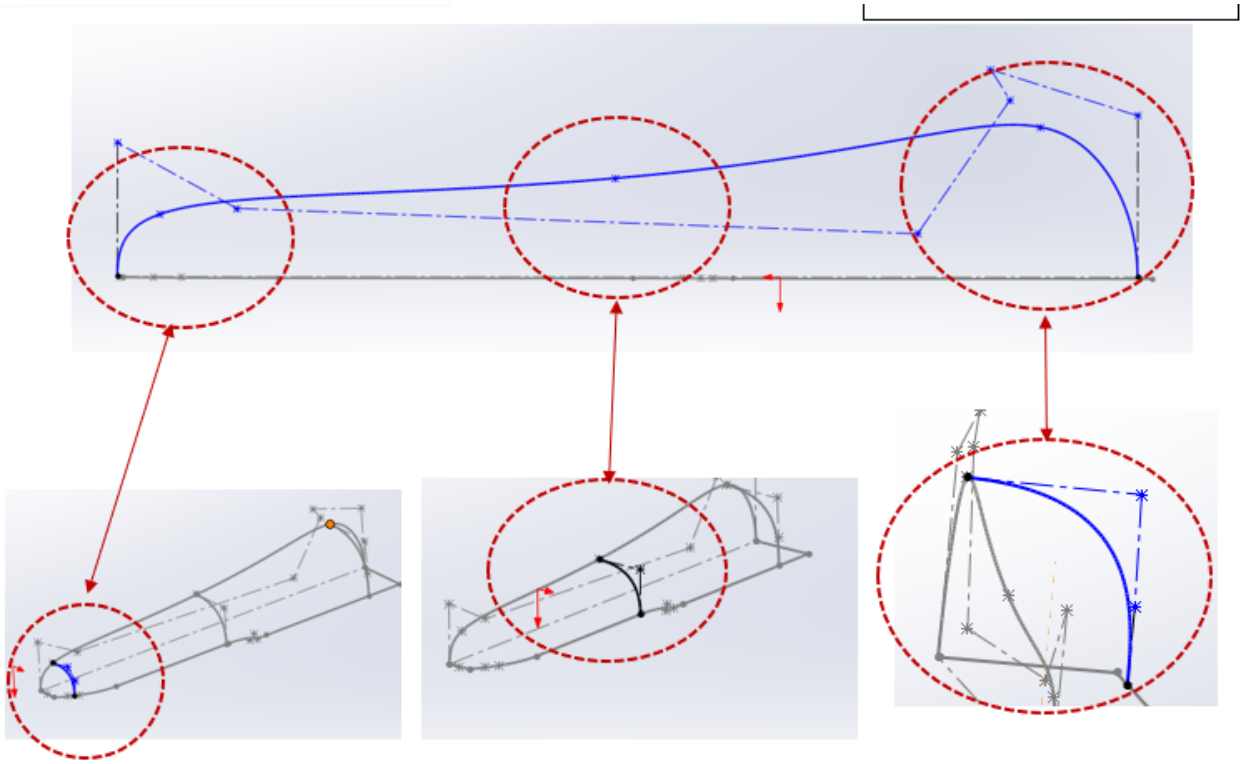
Nous allons maintenant passer à la partie supérieure et tout comme la partie inférieure, nous allons commencer par le croquis 2d de base.


9.50
8

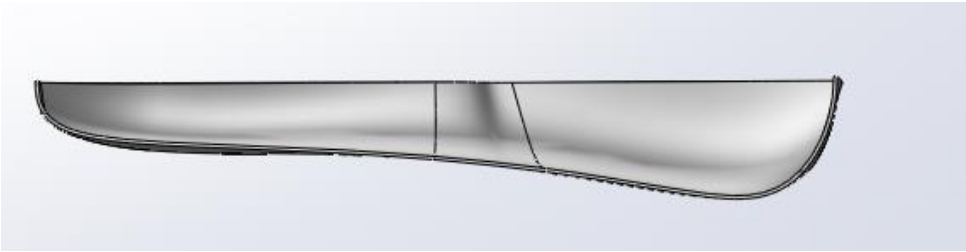


Nous avons suivi la même approche de la partie supérieure, nous avons fait une courbure qui est orientée vers le haut cette fois, nous avons essayé de garder simple et l'avant a dû être plus grand que le reste de la boddy car il tiendra le capteur à ultrasons.

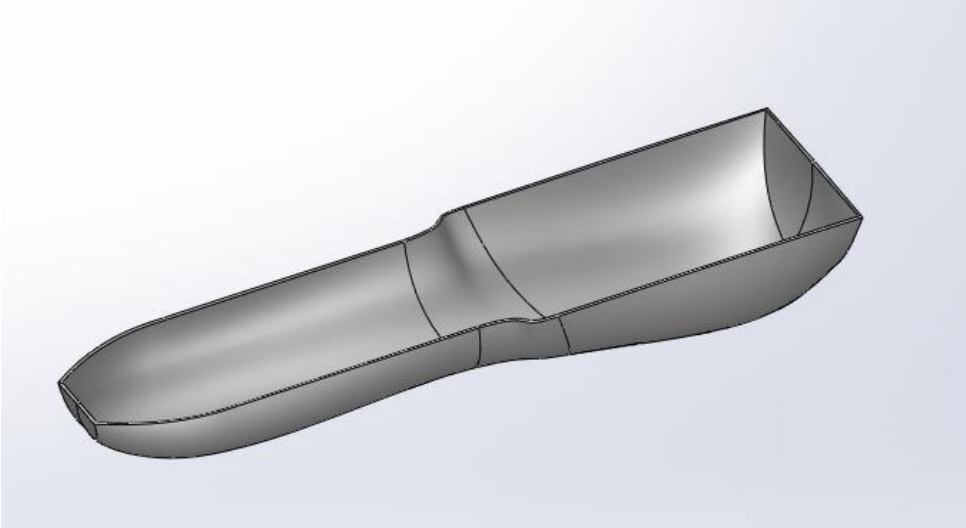
Des courbures latérales ont été faites pour façonner la surface de la pièce.



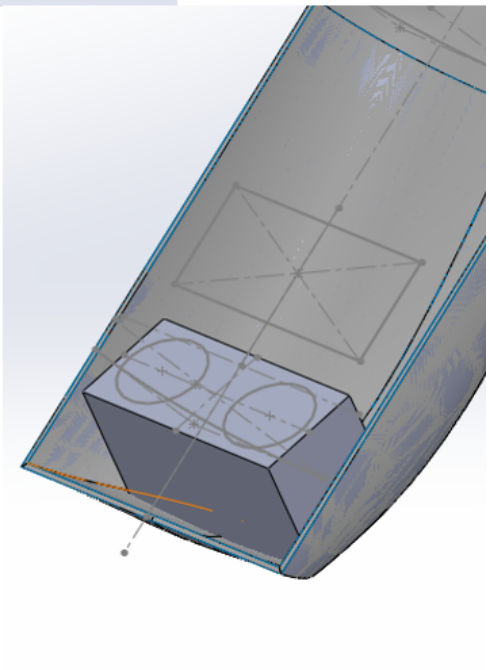
On a fini avec un corps qui ressemble à ça.
Mais un gros problème s'est posé quand nous avons essayé d'y monter des pièces .



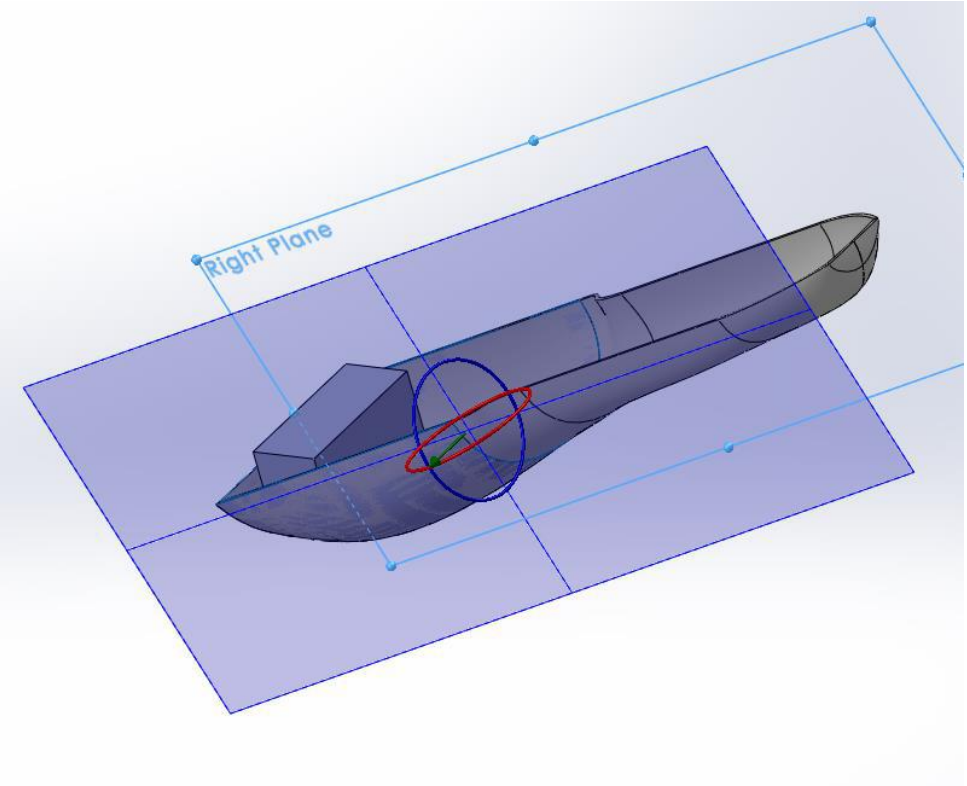
Pour une raison quelconque, cette approche de faire des courbures et de créer une surface d'eux afin que nous puissions avoir un corps solide à la fin ne fonctionnait pas de notre côté comme la partie inférieure.

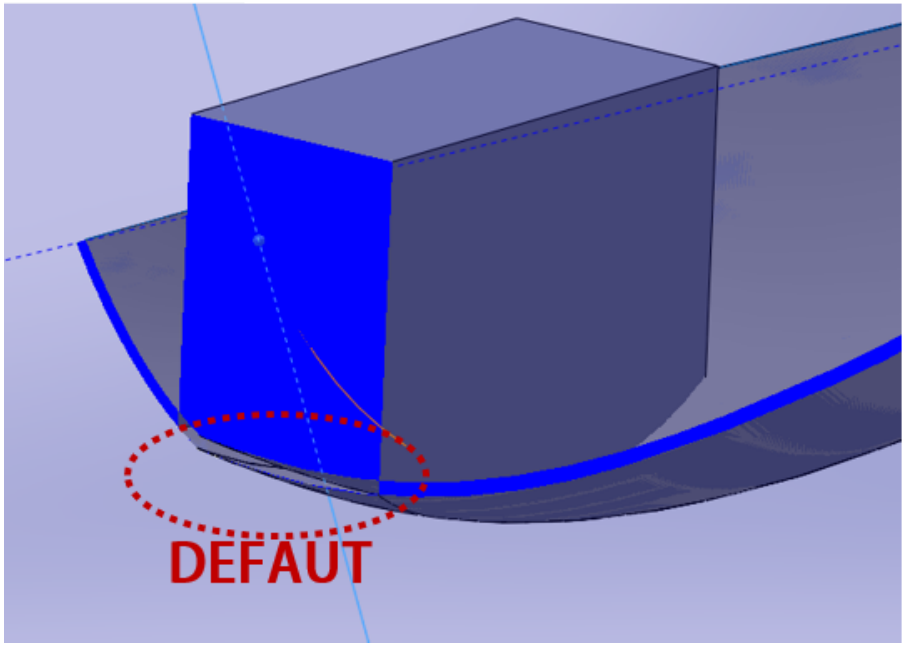


Lorsque nous avons essayé de monter la prise du capteur, il n'a pas réussi à faire le contact complet et fusionner avec la partie inférieure



On fait une coupe latérale et on peut voir l'espace vide entre la douille et le corps de la pièce

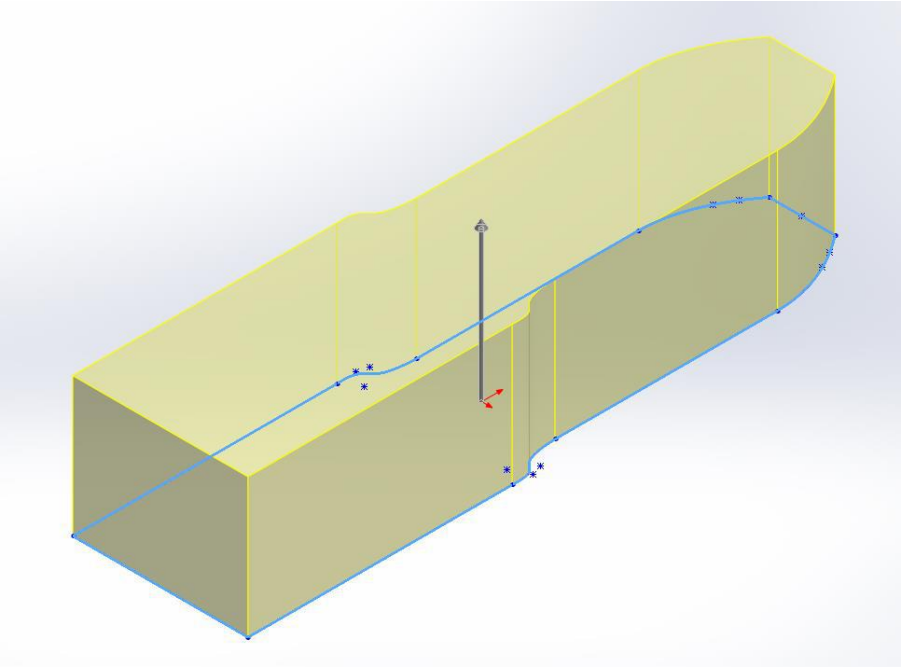




Nous avons essayé de trouver une solution dans la communauté SOLIDWORKS, mais nous n'en avons pas trouvé.

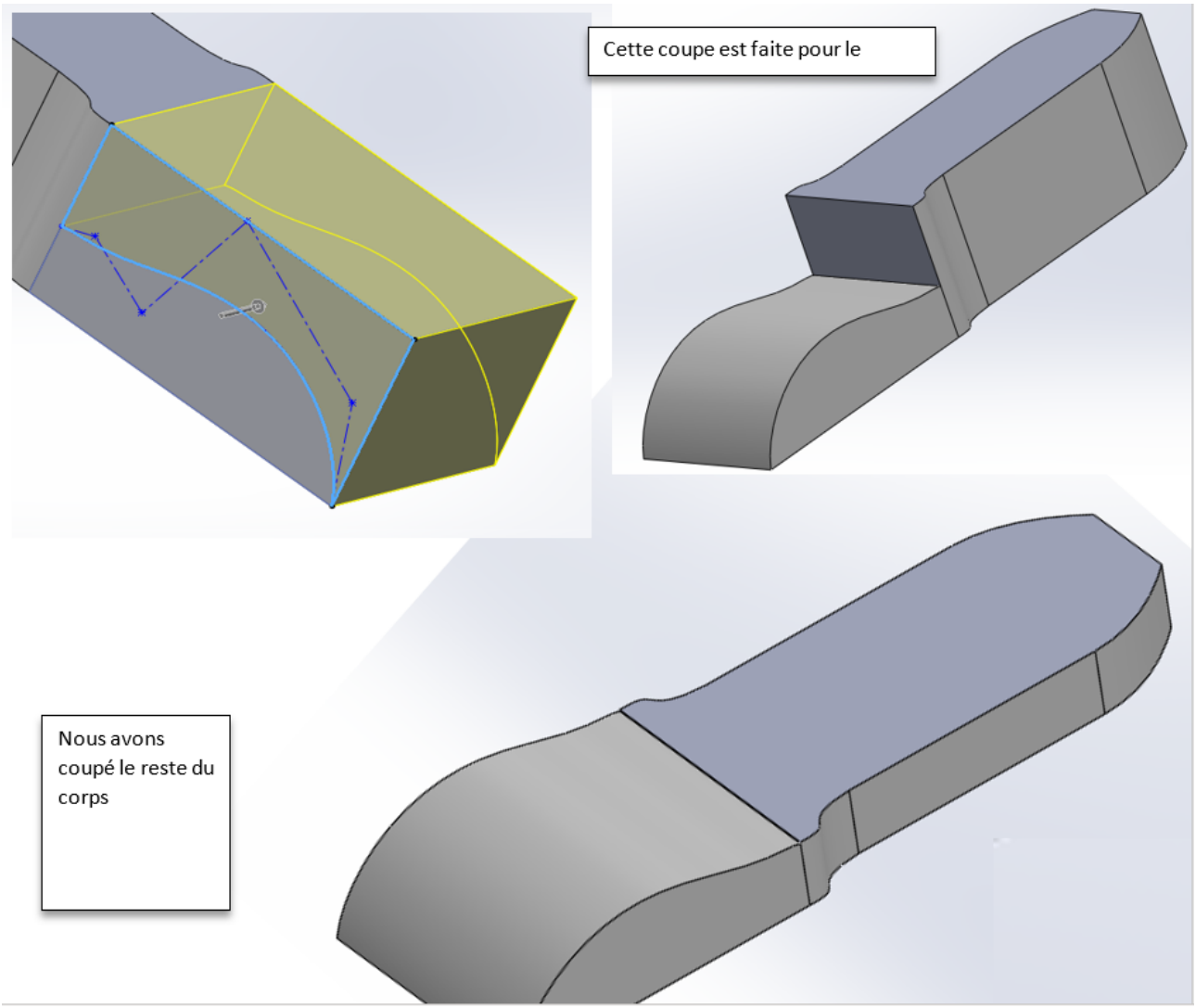
Nous devons changer l'approche et concevoir la partie supérieure d'une autre façon.

Alors au lieu de courbures, faisons un corps solide à partir du croquis de base 2d !

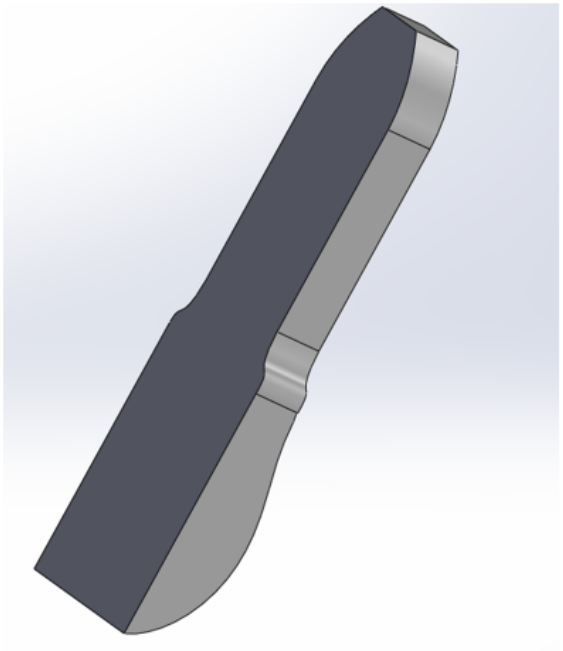


Ensuite, nous allons faire des coupes dans le corps solide jusqu'à ce que nous sommes satisfaits des résultats.

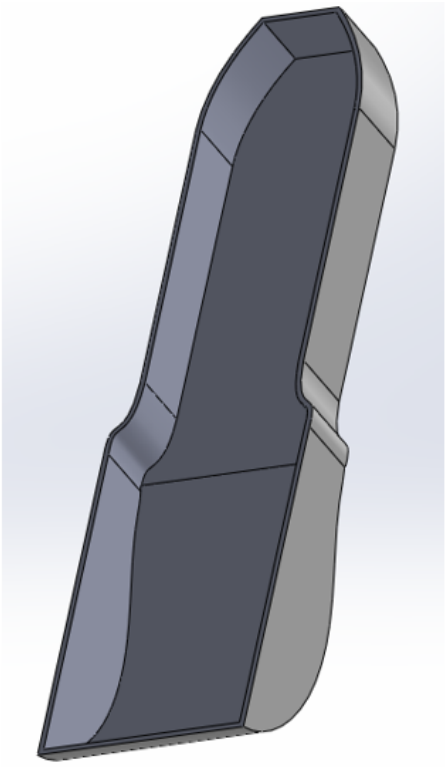
Commençons par la première coupe :


Cette coupe est faite pour le
Nous avons coupé le reste du corps

La pièce était un corps solide nous avons enlevé le matériel à l'intérieur afin que nous puissions monter nos pièces

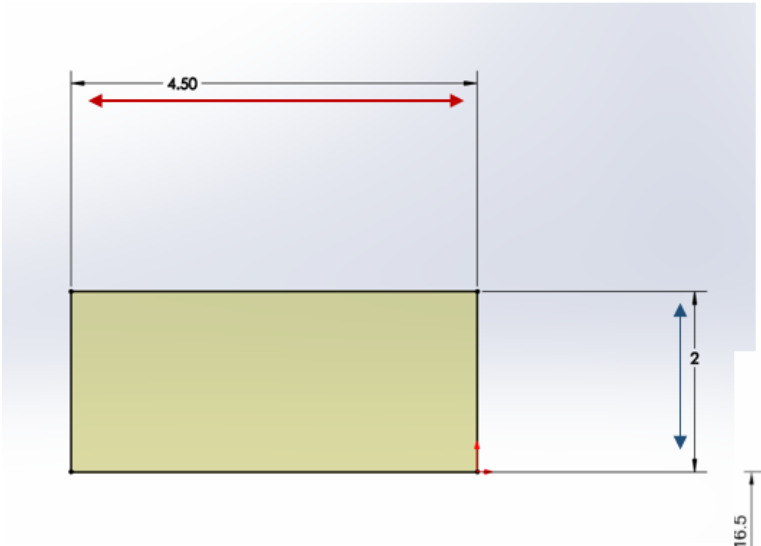


La pièce était un corps solide nous avons enlevé le matériel à l'intérieur afin que nous

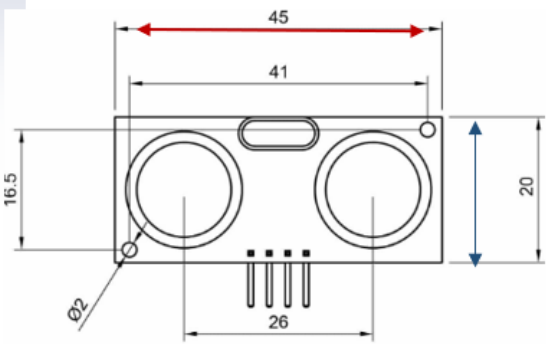


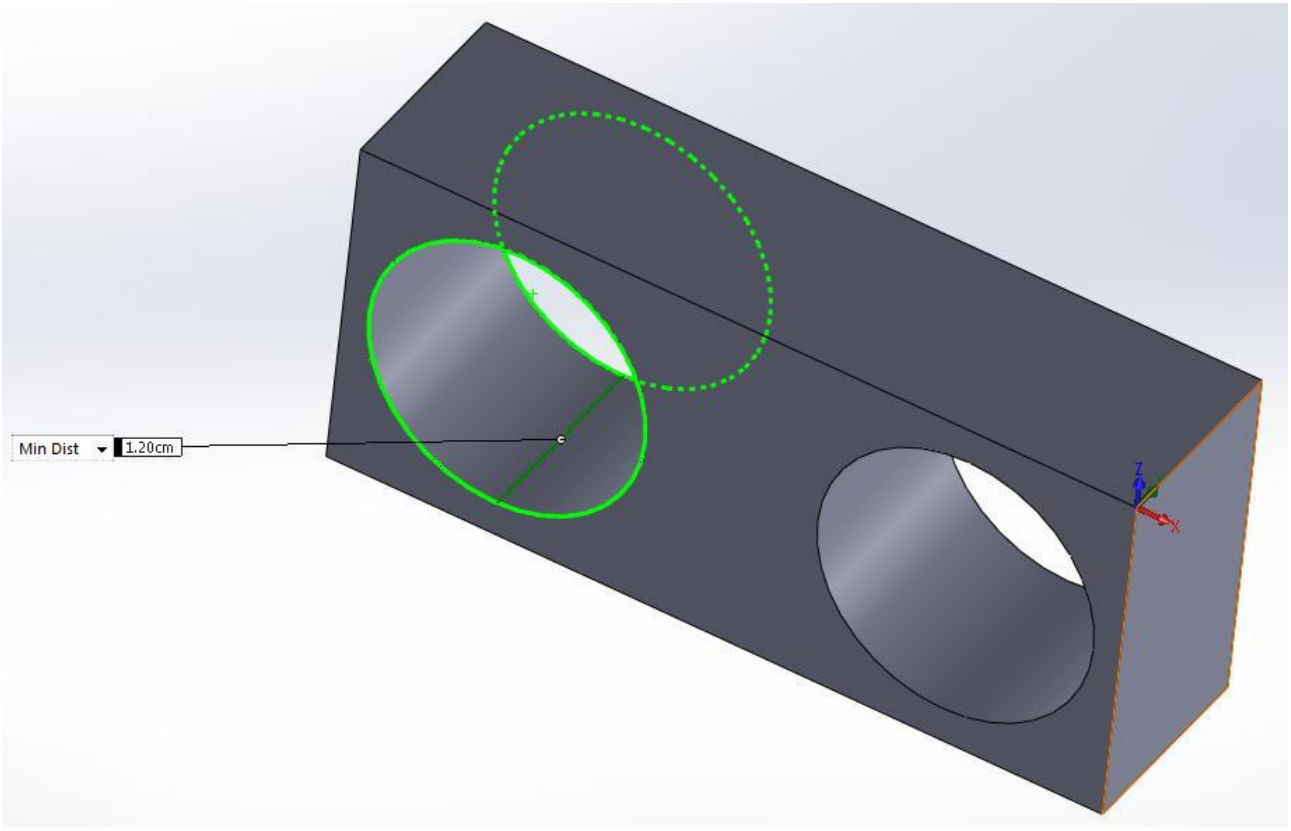
Nous allons maintenant commencer à monter la première partie dans la partie supérieure après nous être assurés qu'il n'y a pas de problèmes comme la partie précédente.

nous avons commencé avec le capteur à ultrasons

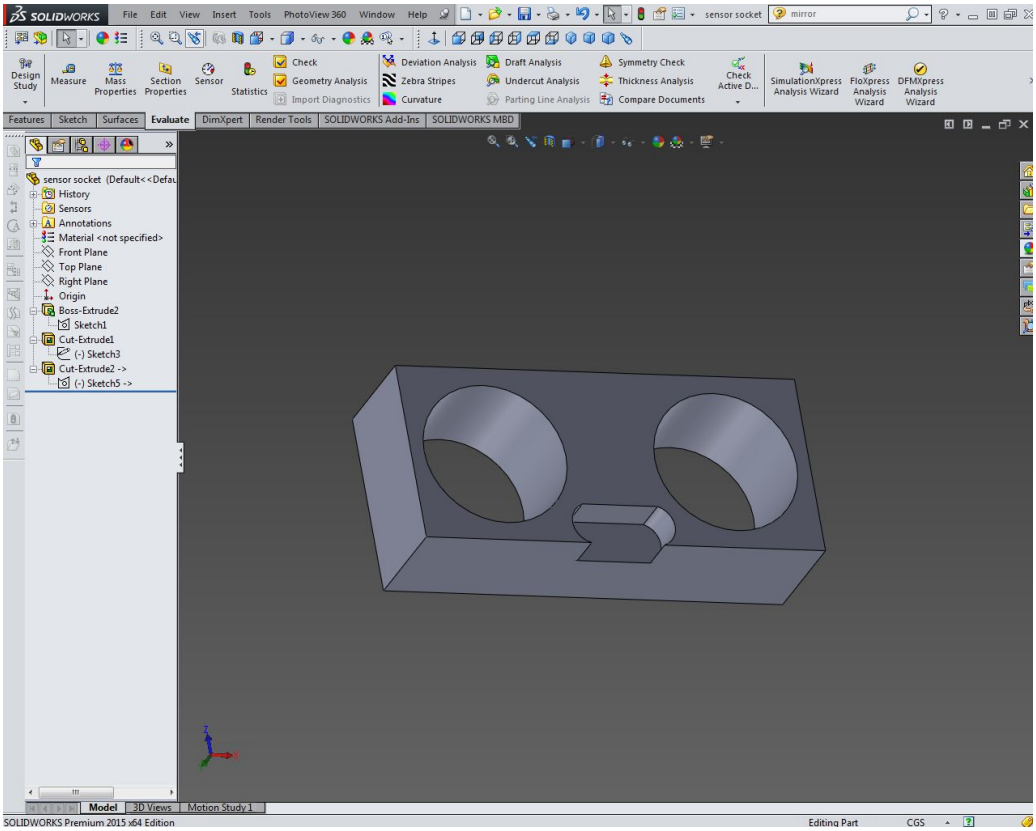


Un croquis en 2d était le point de départ du support de capteur, il doit correspondre aux mesures du capteur

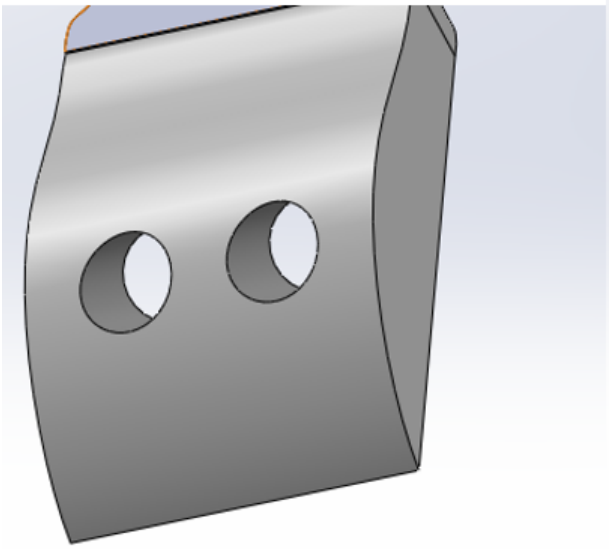


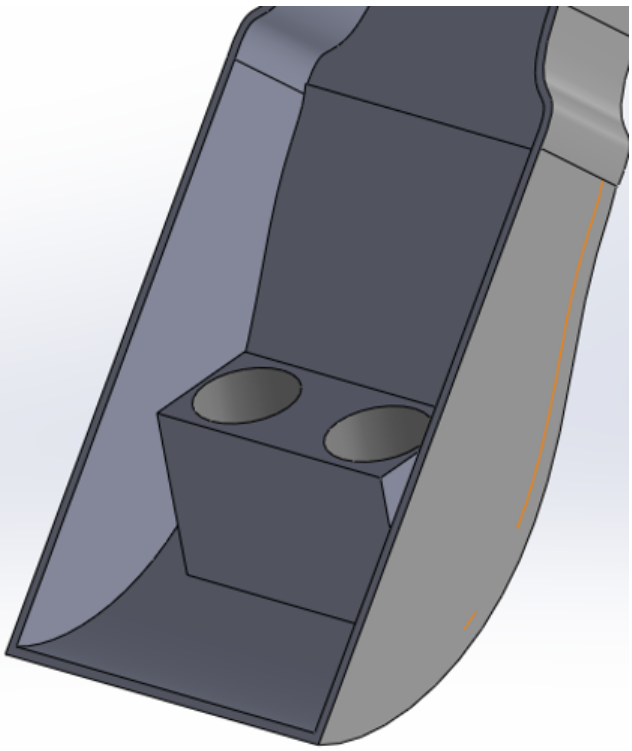


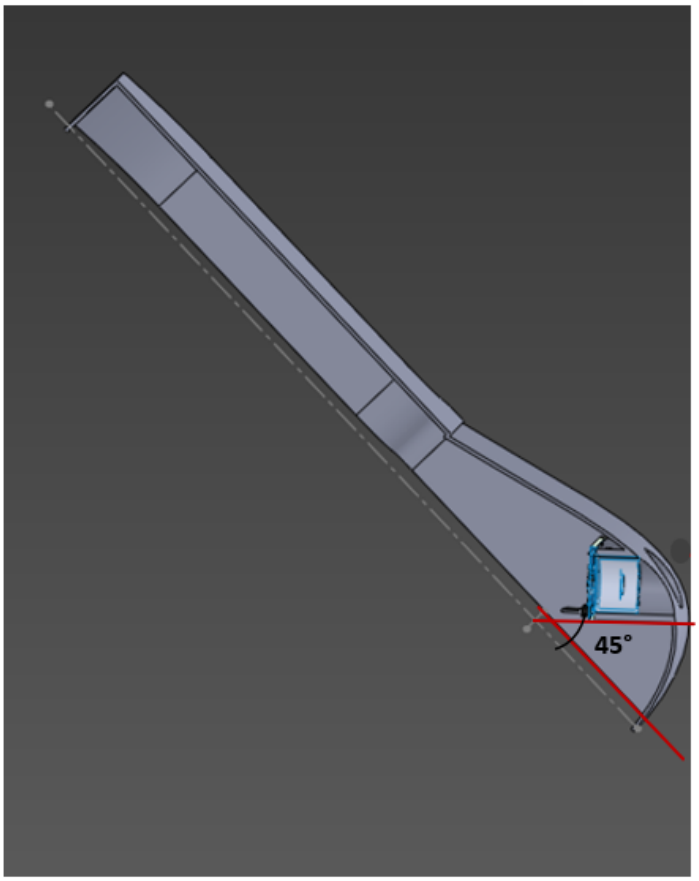
Voici à quoi ressemble le résultat final de la socket. Le capteur HC-SR04 peut maintenant s'insérer parfaitement dans la partie supérieure de la canne



Après l'avoir fini, nous avons dû le fusionner avec la partie supérieure



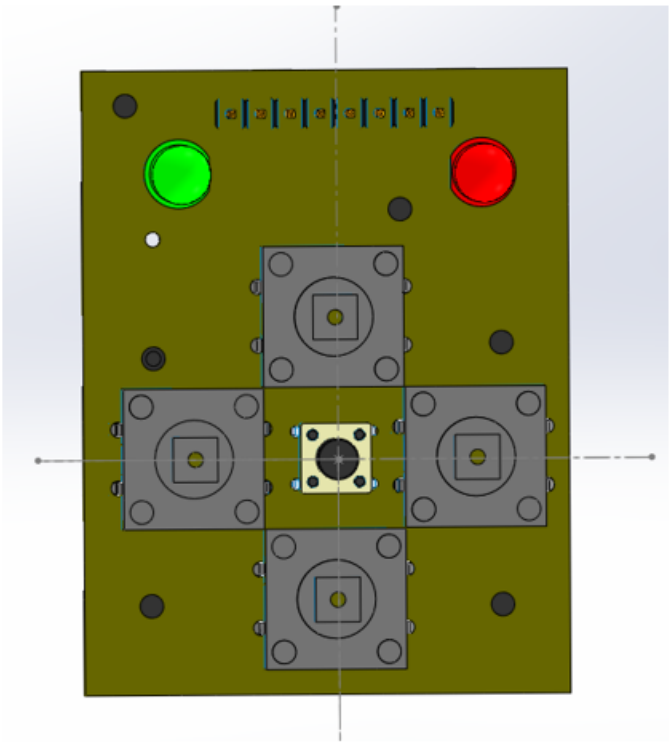




Nous avons monté le capteur pour voir si je vais bien et il l'a fait !

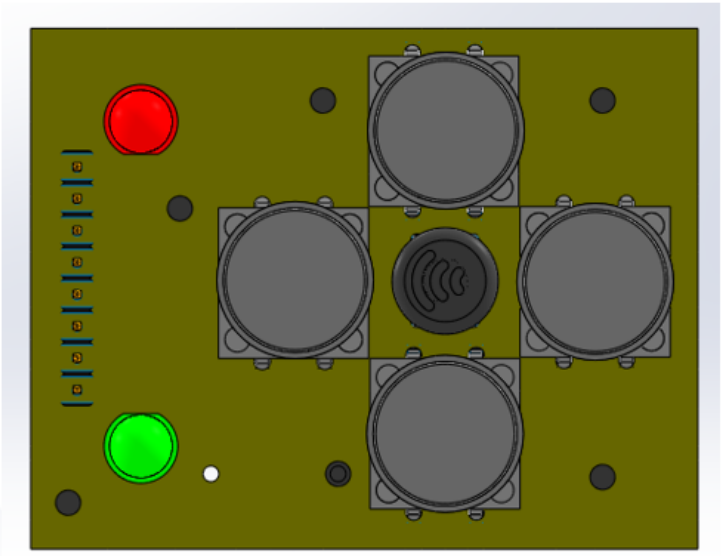
Il a été monté à 45 degrés de sorte que lorsque l'utilisateur va tenir le peut le capteur sera horizontal et pas incliné.

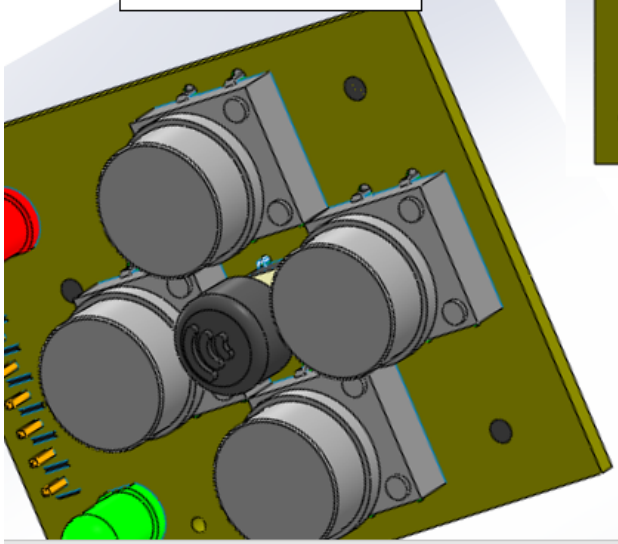
On se déplace ensuite pour monter les boutons pcb et buzzer place.



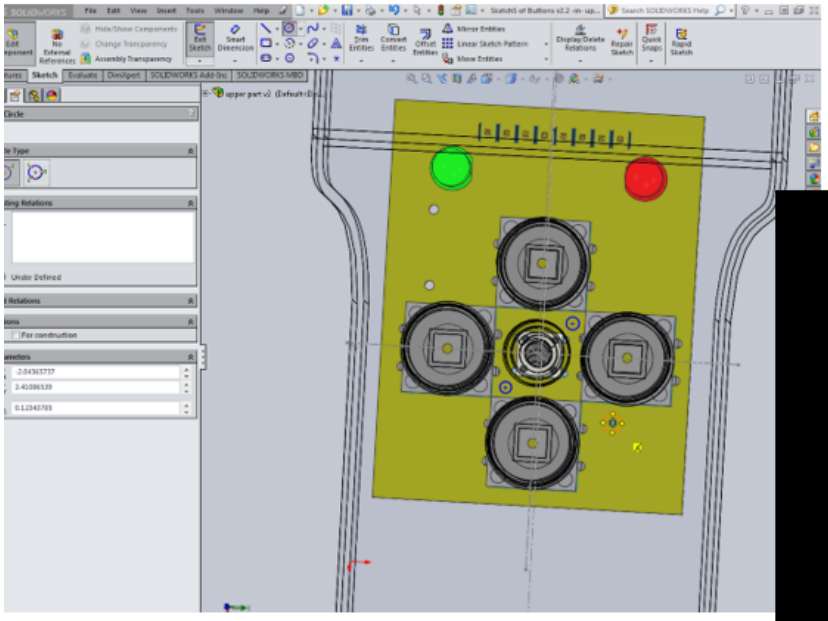
C'est le modèle 3d des boutons PCB, comme nous l'avons mentionné précédemment les boutons latéraux ont leurs propres bouchons mais celui au milieu nous voulions qu'il soit unique donc nous l'avons fait !

Après avoir monté tous les bouchons, nous nous retrouvons avec ce résultat.



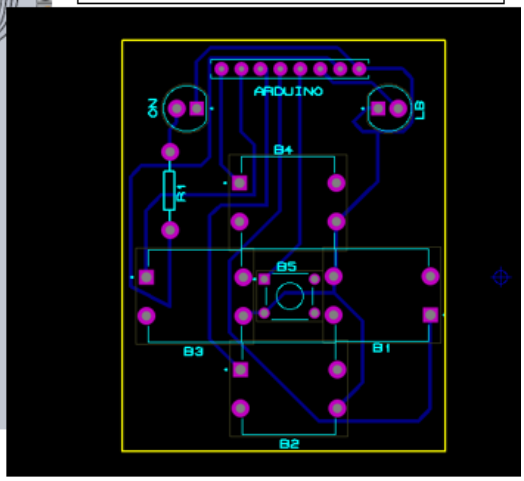


Nous sommes passés à la partie supérieure, et tout comme la carte Arduino, l'autre circuit doit être monté et vissé sur la partie supérieure.



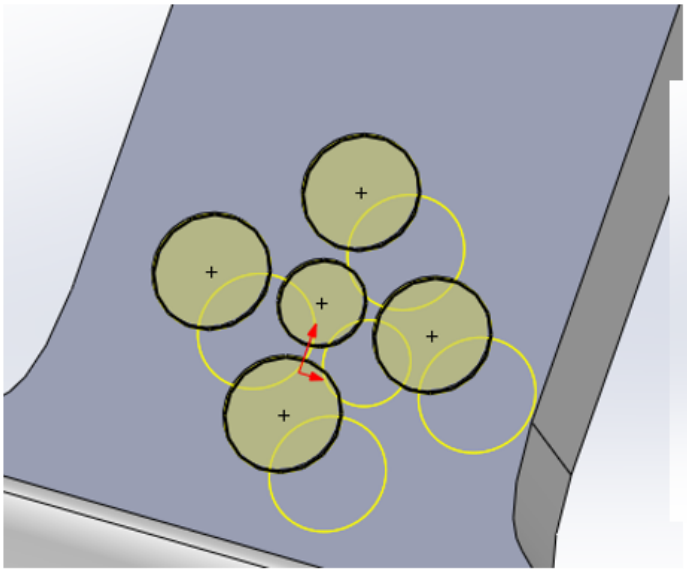
Nous avons fait des trous dans le circuit imprimé pour le monter sur la canne en prenant en considération les chemins de connexion dans le circuit imprimé.

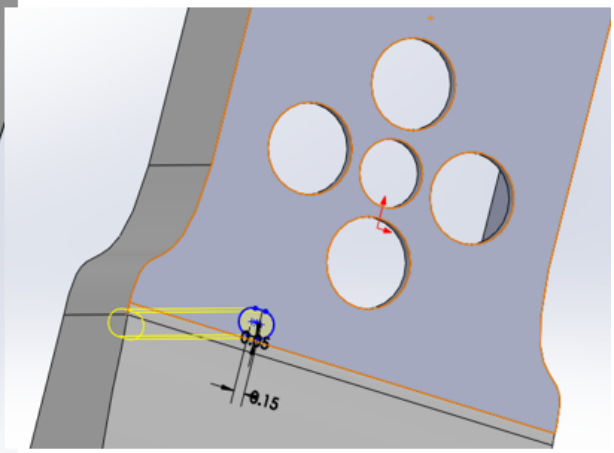
Nous les avons aussi faites dans la partie supérieure.



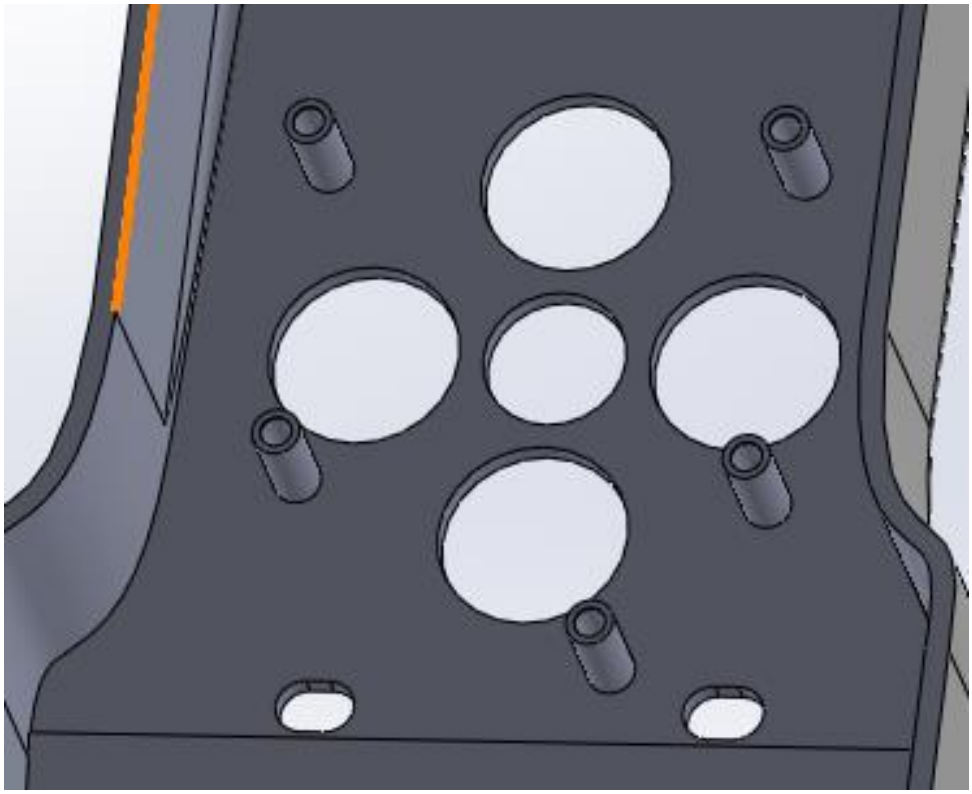
Nous avons fait les trous de bouchons, le diamètre devait être légèrement plus grand que le diamètre des bouchons afin que nous puissions avoir un peu de mouvement pendant que l'utilisateur clique sur les

Après avoir fait les boutons trous, nous avons fait deux autres trous sur le





Voilà à quoi tout ressemblait !

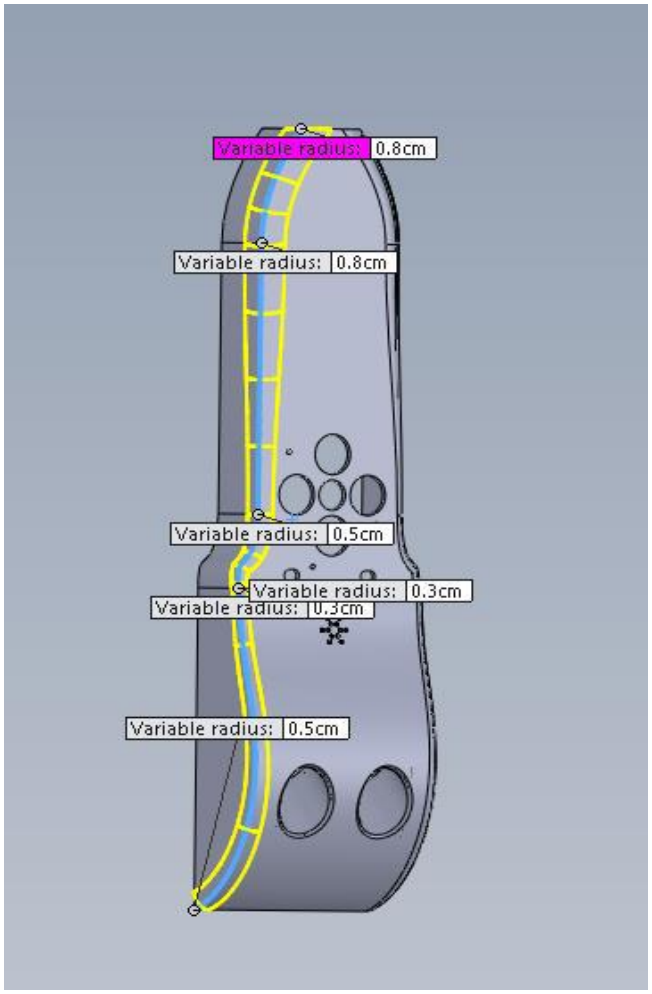


Nous avons ensuite fait une place pour le buzzer avec des trous de sorte que les sons peuvent être entendus clairement, nous avons ajouté deux places pour les moteurs sur les deux côtés de la partie supérieure des murs.

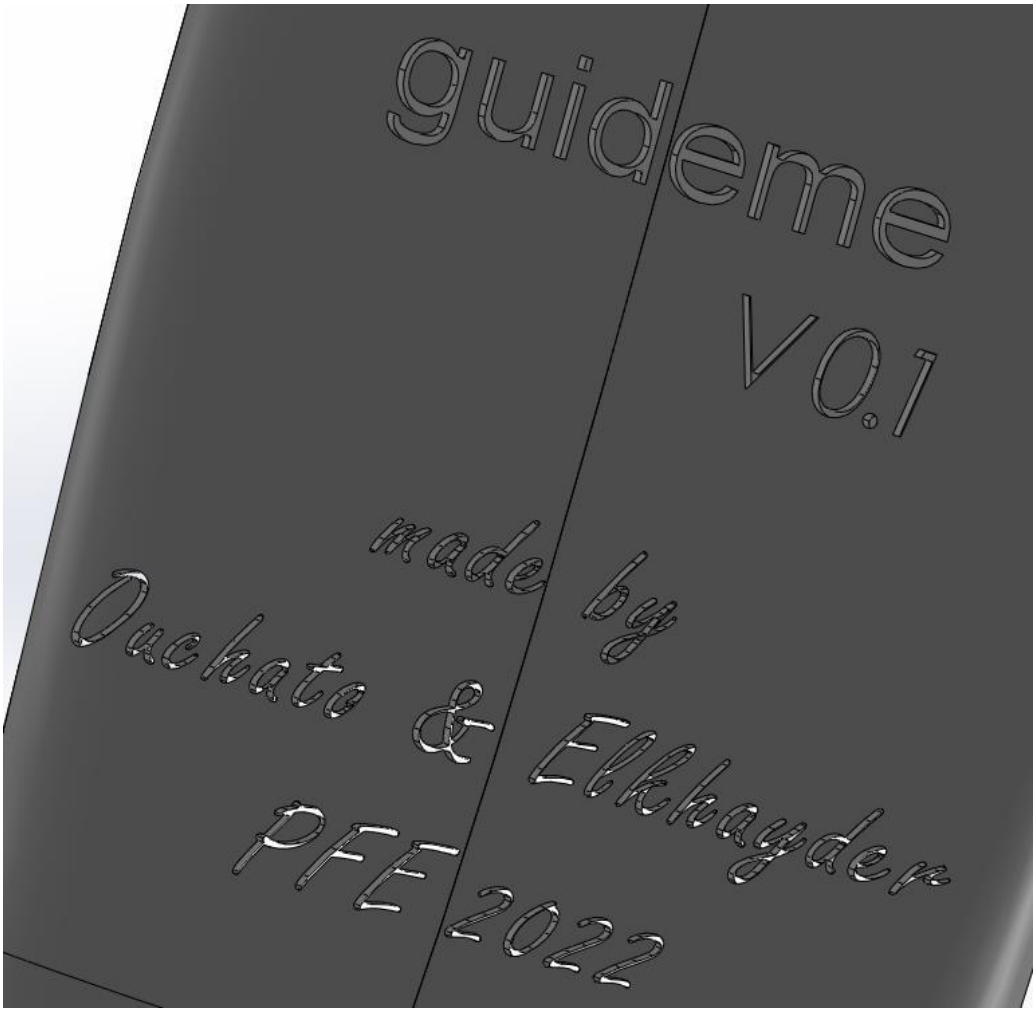
Et finalement, nous avons fait une place pour la LED flash et un endroit où wo va monter le circuit de charge.

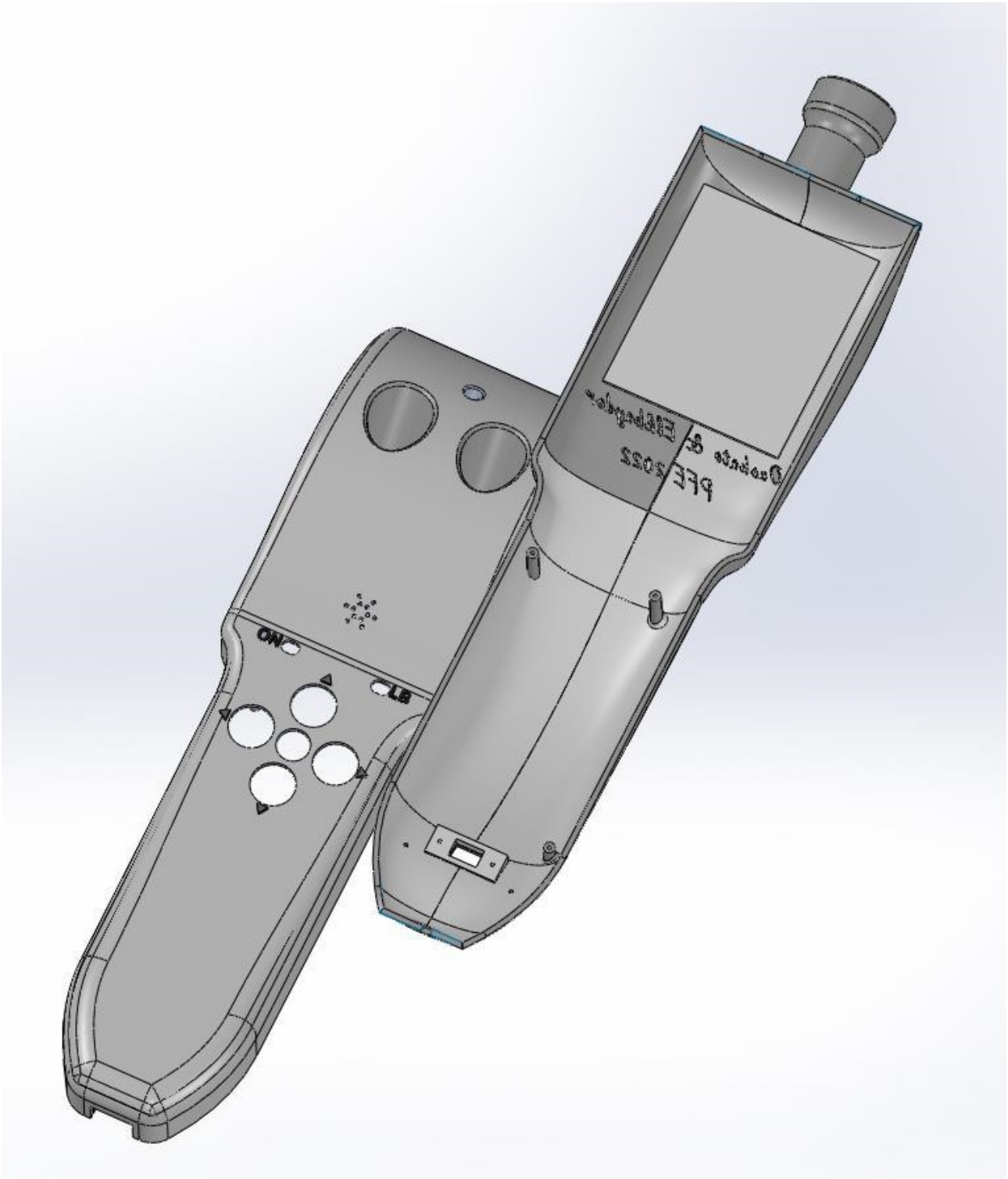
Les bords de la partie supérieure étaient trop pointus et agressifs pour ce type de produit, la canne devait être lisse

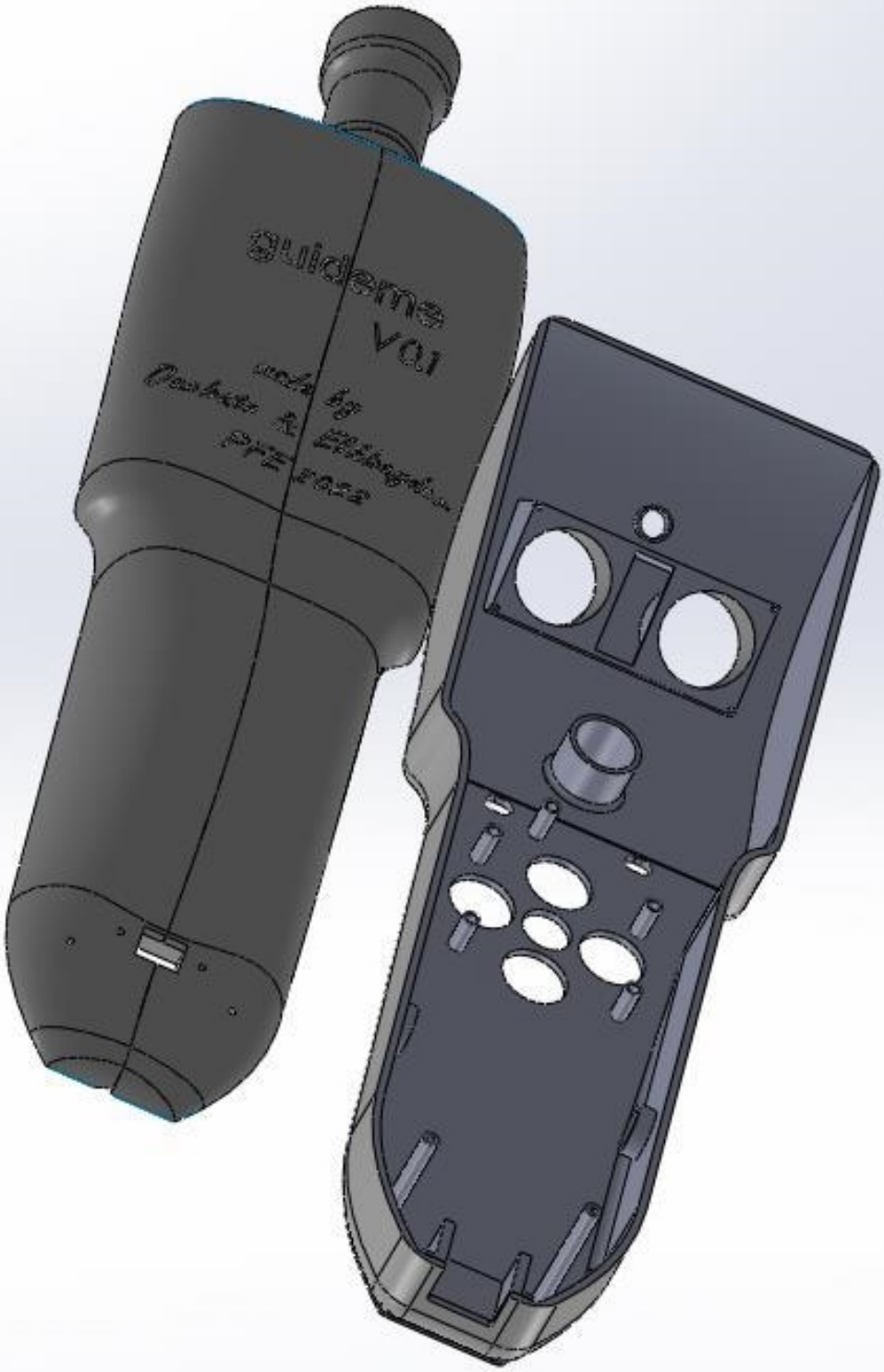
Il manquait aussi un caractère, nous avons donc ajouté quelques détails.



Chaque produit a un nom au dos ou sur l'un de ses côtés, il se présente à celui qui le porte, notre produit n'a pas de nom aucune identification, nous avons dû le changer donc nous avons gravé au dos de la partie inférieure le nom que nous avons trouvé : « guideme » Nous avons également inclus nos noms en tant que fondateurs de ces produits et le contexte de sa réalisation qui est le projet de dernière année.


guideme
V0.1
made by
Ouchato & Elkhayder
PFE 2022


ON
LB


Sidema
V01

3 | Finalisation et réalisation de la canne

3.1 Les étapes de la réalisation d'un circuit imprimé

3.1.1 Découpe

Découpe à la cisaille d'une plaque de la taille du circuit à réaliser, suppression de l'adhésif de protection.

Cisaille guillotine

Cisaille guillotine de capacité de coupe de 380mm équipée de lames en acier spécial étudiées pour la coupe des matériaux employés dans la fabrication des circuit imprimes (XXXF, VERRE EPOXY). Cette machine peut-être aussi utlisee pour la coupe des materiaux comme :

- L'aluminium
- Cuivre
- Laiton
- Acier
- ...

Cette machine est équipée d'une butée d'angle et d'une butée arrière réglable pour découpe en série.

Caractéristiques techniques

- Largeur de coupe : 380mm
- Longueur butée arrière : 175mm
- Dimensions : 620 x 430 x 450 mm
- Poids : 45Kg

Capacité de coupe

- Tète acier doux : 1.2mm

— Tète aluminium : 1.5mm

— Stratifiés : 1.6mm

3.1.2 Insolation

Positionnement de la plaque et du calque dans l'insoleuse. Ils sont maintenus par le vide pendant l'exposition aux rayons ultras-violets

Durée d'insolation 3mn

FIGURE 3.1 – Châssis d'insolation

3.1.3 Immersion

Immersion de la plaque pendant 2 min dans bain de révélateur.

Rinçage abondant à l'eau courante pour stopper l'action du révélateur.

Seules les parties non exposées aux U.V restent protégées par la couche photosensible.

Le révélateur utilisé (hydrate de soude) est livré en granules.

Dissolver le contenu d'un sachet dans un litre d'eau (max : 25°C).

Il doit toujours être remis immédiatement en bouteille après utilisation (Altération à l'air).

3.1.4 Gravure

Par projection de perchlorure de fer pour éliminer le cuivre non protégée. Durée de gravure environ 5 minutes.

Rinçage très abondant de l'eau courant.

— Attention : le perchlorure de fer est un produit dangereux (acide)

— Attention : aux projections (yeux, habits, ...). Mettez une blouse et des lunettes de protection !

FIGURE 3.2 – Graveuse CIF bb4

3.1.5 Nettoyage

Élimination de la couche de protection avec un tampon imbibé d'alcool.

3.1.6 Perçage

Forêt carbure a partir de diamètre 0.5mm. Vitesse de rotation : 15000 à 20000 Tr/min

Forêt acier rapide à partir de diamètre 0.8mm. Vitesse de rotation : 2250 à 3000 Tr/min

FIGURE 3.3 – Perceuse

3.2 PCB du circuit des boutons

Après avoir faire la conception du circuit des boutons (Chapitre 2.7.2.4), on doit maintenant le réaliser suivant les étapes expliquées précédemment (Chapitre 3.1).

Après la découpe, l'insolation et (on manque les photos), on l'immerge :

FIGURE 3.4 – L'immersion du circuit imprimé des boutons

Puis on fait le perçage :

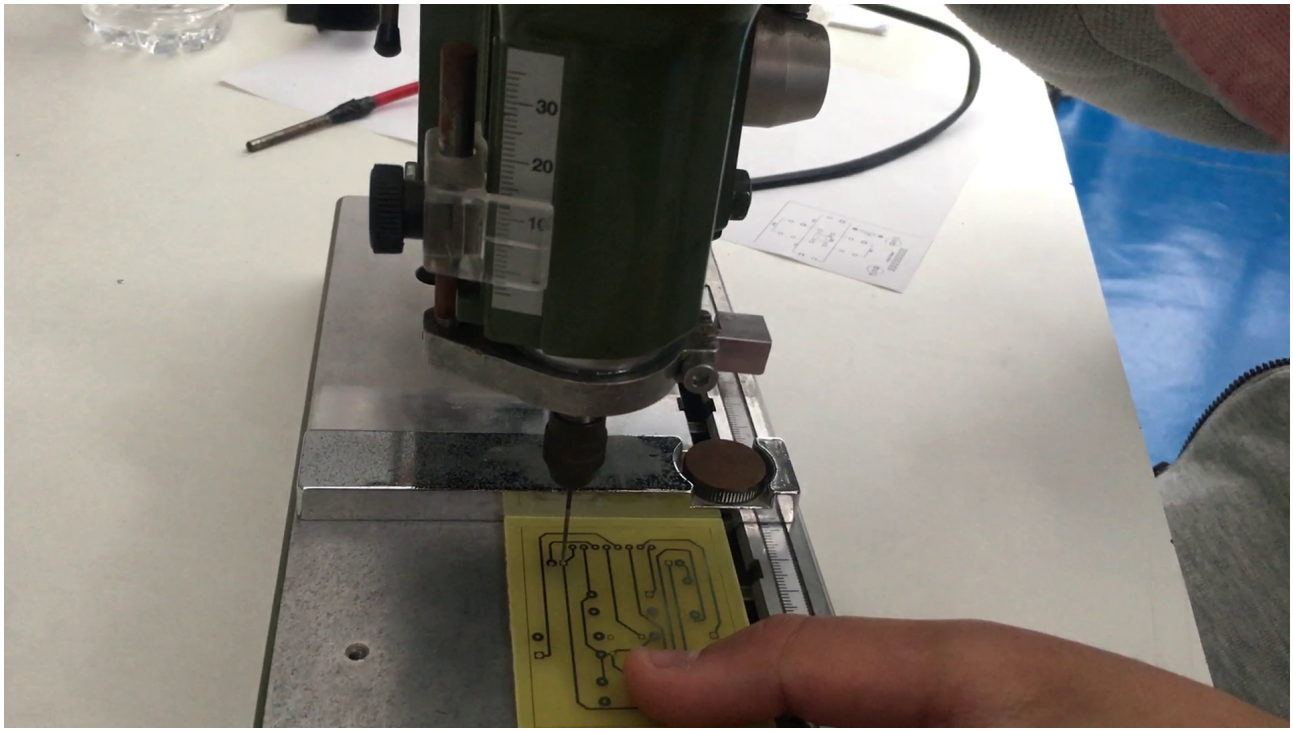


FIGURE 3.5 – Perçage du circuit imprimé des boutons

Après, nous soudons les composant sur la carte :

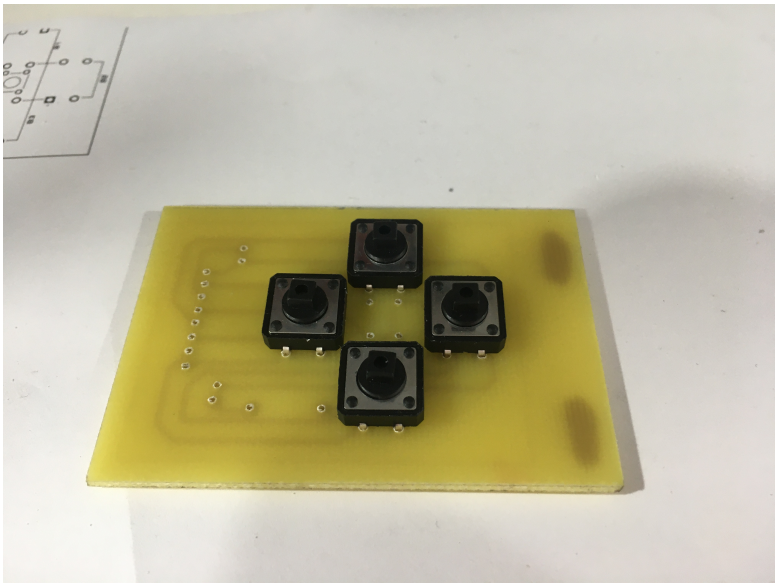


FIGURE 3.6 – Boutons soudés circuit imprimé des boutons

Carte des boutons final :

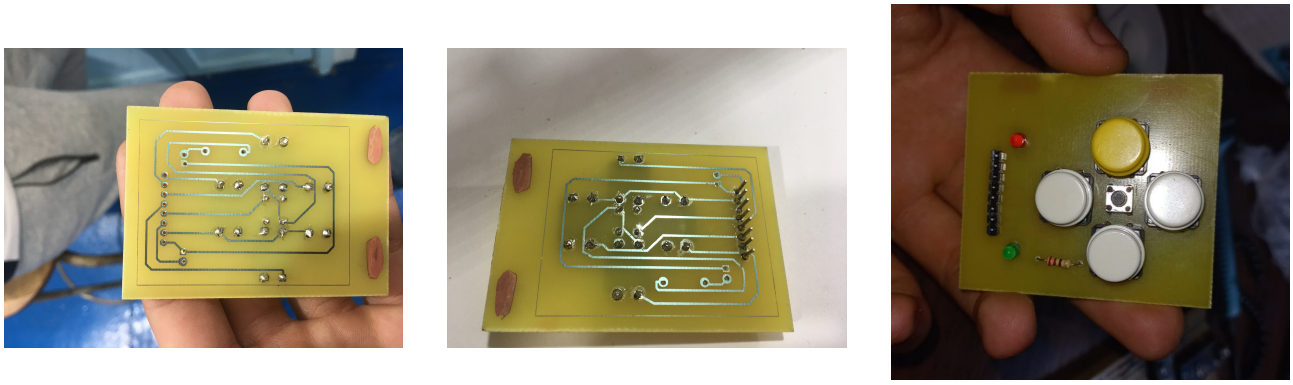


FIGURE 3.7 – Circuit imprimé final des boutons

3.3 PCB du circuit Arduino

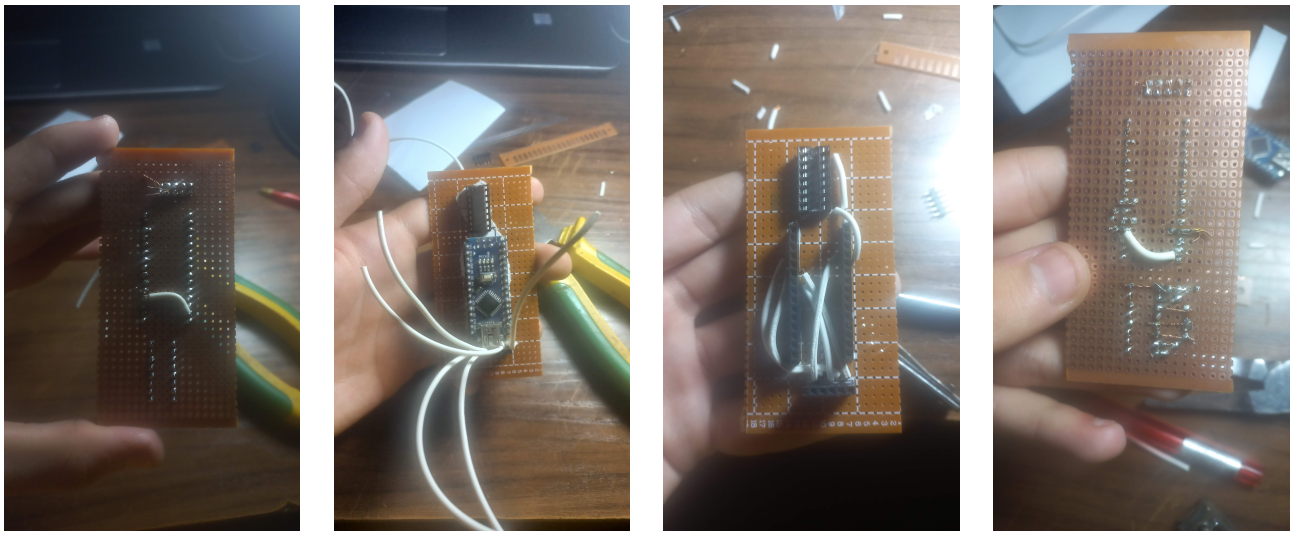
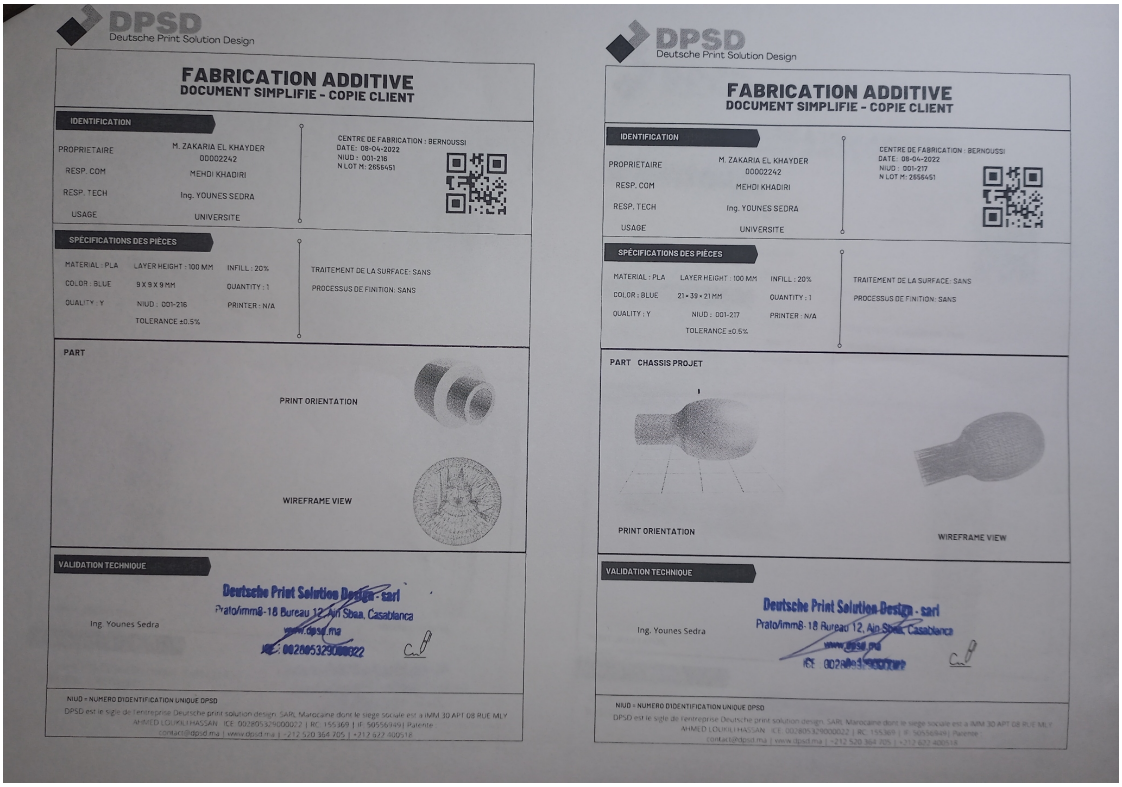


FIGURE 3.8 – La réalisation du circuit d'Arduino

3.4 L'impression 3D

L'impression du carcasse de la canne est sous-traitée chez la société **Deutsche Print Solution Design**



DPSD
Deutsche Print Solution Design

FABRICATION ADDITIVE
DOCUMENT SIMPLIFIE - COPIE CLIENT

IDENTIFICATION

PROPRIETAIRE	M. ZAKARIA EL KHAYDER 00002242
RESP. COM	MEHDI KHADIRI
RESP. TECH	Ing. YOUNES SEDRA
USAGE	UNIVERSITE

CENTRE DE FABRICATION : BERNOUSSI
DATE : 08-04-2022
NIUD : 001-216
N LOT M: 2658451

SPÉCIFICATIONS DES PIÈCES

MATERIAL : PLA	LAYER HEIGHT : 100 MM	INFILL : 20%	TRAITEMENT DE LA SURFACE : SANS
COLOR : BLUE	9 X 9 X 9 MM	QUANTITY : 1	PROCESSUS DE FINITION : SANS
QUALITY : Y	NIUD : 001-216	PRINTER : N/A	
	TOLERANCE ±0.5%		

PART

PRINT ORIENTATION

WIREFRAME VIEW

VALIDATION TECHNIQUE

Ing. Younes Sedra

Deutsche Print Solution Design - sarl
Prato/imm8-18 Bureau 12, Ain Sbaa, Casablanca
www.dpsd.ma
ICE : 002805329000022

NIUD = NUMERO D'IDENTIFICATION UNIQUE DPSD
DPSD est le sigle de l'entreprise Deutsche print solution design, SARL Marocaine dont le siege sociale est a IMM 30 APT 08 RUE MLY AHMED LOUKILI HASSAN ICE: 002805329000022 | RC: 155369 | IF: 50556949 | Patente
contact@dpsd.ma | www.dpsd.ma | +212 520 364 705 | +212 627 400518

DPSD
Deutsche Print Solution Design

FABRICATION ADDITIVE
DOCUMENT SIMPLIFIE - COPIE CLIENT

IDENTIFICATION

PROPRIETAIRE	M. ZAKARIA EL KHAYDER 00002242
RESP. COM	MEHDI KHADIRI
RESP. TECH	Ing. YOUNES SEDRA
USAGE	UNIVERSITE

CENTRE DE FABRICATION : BERNOUSSI
DATE : 08-04-2022
NIUD : 001-217
N LOT M: 2658451

SPÉCIFICATIONS DES PIÈCES

MATERIAL : PLA	LAYER HEIGHT : 100 MM	INFILL : 20%	TRAITEMENT DE LA SURFACE : SANS
COLOR : BLUE	21 × 39 × 21 MM	QUANTITY : 1	PROCESSUS DE FINITION : SANS
QUALITY : Y	NIUD : 001-217	PRINTER : N/A	
	TOLERANCE ±0.5%		

PART CHASSIS PROJET

PRINT ORIENTATION

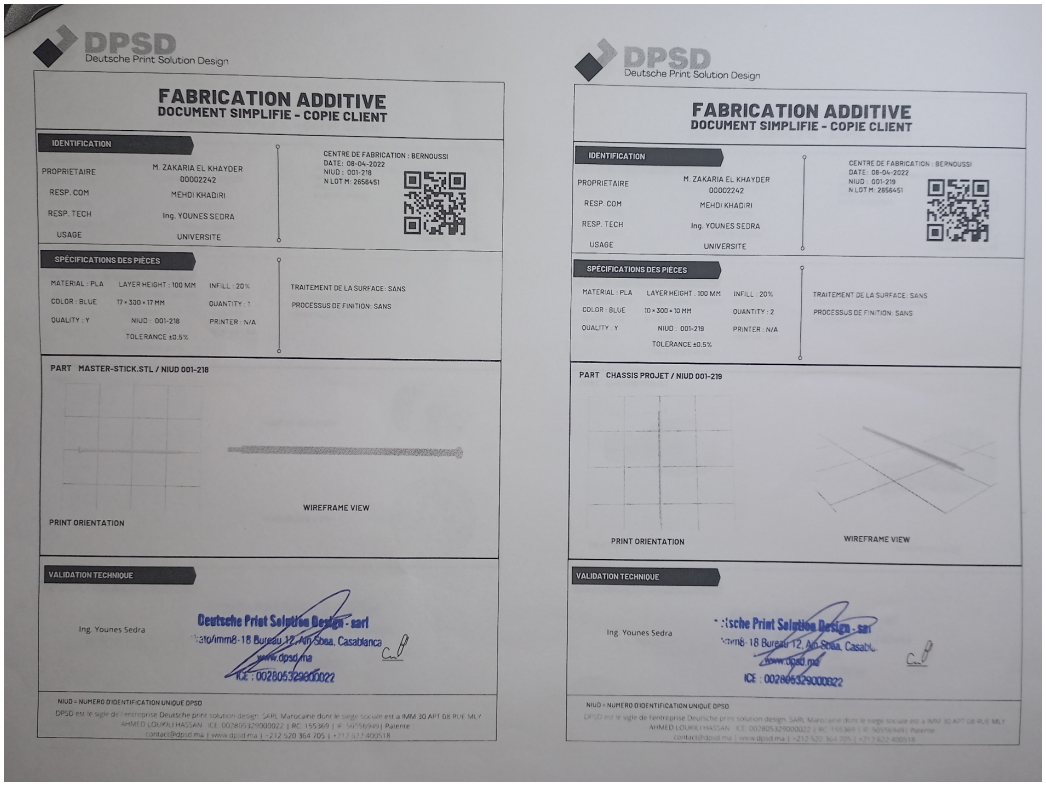
WIREFRAME VIEW

VALIDATION TECHNIQUE

Ing. Younes Sedra

Deutsche Print Solution Design - sarl
Prato/imm8-18 Bureau 12, Ain Sbaa, Casablanca
www.dpsd.ma

NIUD = NUMERO D'IDENTIFICATION UNIQUE DPSD
DPSD est le sigle de l'entreprise Deutsche print solution design, SARL Marocaine dont le siege sociale est a IMM 30 APT 08 RUE MLY AHMED LOUKILI HASSAN ICE: 002805329000022 | RC: 155369 | IF: 50556949 | Patente
contact@dpsd.ma | www.dpsd.ma | +212 520 364 705 | +212 627 400518



DPSD
Deutsche Print Solution Design

FABRICATION ADDITIVE
DOCUMENT SIMPLIFIE - COPIE CLIENT

IDENTIFICATION

PROPRIETAIRE	M. ZAKARIA EL KHAYDER 00002242
RESP. COM	MEHDI KHADIRI
RESP. TECH	Ing. YOUNES SEDRA
USAGE	UNIVERSITE

CENTRE DE FABRICATION : BERNOUSSI
DATE : 08-04-2022
NIUD : 001-218
N LOT M: 2658451

SPÉCIFICATIONS DES PIÈCES

MATERIAL : PLA	LAYER HEIGHT : 100 MM	INFILL : 20%	TRAITEMENT DE LA SURFACE : SANS
COLOR : BLUE	17 × 320 × 17 MM	QUANTITY : 1	PROCESSUS DE FINITION : SANS
QUALITY : Y	NIUD : 001-218	PRINTER : N/A	
	TOLERANCE ±0.5%		

PART MASTER-STICK.STL / NIUD 001-218

WIREFRAME VIEW

PRINT ORIENTATION

VALIDATION TECHNIQUE

Ing. Younes Sedra

Deutsche Print Solution Design - sarl
Prato/imm8-18 Bureau 12, Ain Sbaa, Casablanca
www.dpsd.ma
ICE : 002805329000022

NIUD = NUMERO D'IDENTIFICATION UNIQUE DPSD
DPSD est le sigle de l'entreprise Deutsche print solution design, SARL Marocaine dont le siege sociale est a IMM 30 APT 08 RUE MLY AHMED LOUKILI HASSAN ICE: 002805329000022 | RC: 155369 | IF: 50556949 | Patente
contact@dpsd.ma | www.dpsd.ma | +212 520 364 705 | +212 627 400518

DPSD
Deutsche Print Solution Design

FABRICATION ADDITIVE
DOCUMENT SIMPLIFIE - COPIE CLIENT

IDENTIFICATION

PROPRIETAIRE	M. ZAKARIA EL KHAYDER 00002242
RESP. COM	MEHDI KHADIRI
RESP. TECH	Ing. YOUNES SEDRA
USAGE	UNIVERSITE

CENTRE DE FABRICATION : BERNOUSSI
DATE : 08-04-2022
NIUD : 001-219
N LOT M: 2658451

SPÉCIFICATIONS DES PIÈCES

MATERIAL : PLA	LAYER HEIGHT : 100 MM	INFILL : 20%	TRAITEMENT DE LA SURFACE : SANS
COLOR : BLUE	10 × 300 × 10 MM	QUANTITY : 2	PROCESSUS DE FINITION : SANS
QUALITY : Y	NIUD : 001-219	PRINTER : N/A	
	TOLERANCE ±0.5%		

PART CHASSIS PROJET / NIUD 001-219

PRINT ORIENTATION

WIREFRAME VIEW

VALIDATION TECHNIQUE

Ing. Younes Sedra

ICE : 002805329000022

NIUD = NUMERO D'IDENTIFICATION UNIQUE DPSD

FIGURE 3.9 – Fiches de la fabrication additive

3.5 Le coût du projet

Article	Nombre	Prix total
Pilote des moteurs L293	2	35DH
Les boutons	6	27DH
Les broches	27	45DH
Les fils	40	45DH
Plaque de soudage	1	12DH
Cables	1	8DH
L'impression 3D	-	420DH

TABLE 3.1 – Les charges pour la réalisation du projet

Coût total du projet : 592DH

Webographie

[1] L'organisation Mondiale de la Santé (WHO). *Cécité et déficience visuelle*. URL : https://www.who.int/fr/news-room/fact-sheets/detail/blindness-and-visual-impairment. (Accédé le 05/04/2022).

[2] IOVS | ARVO Journals. *Global Prevalence of Blindness and Distance and Near Vision Impairment in 2020: progress towards the Vision 2020 targets and what the future holds.* URL : https://iovs.arvojournals.org/article.aspx?articleid=2767477. (Accédé le 05/04/2022).

[3] PiBorg. *Ultrasonic Distance Sensor (HC-SR04)*. URL : https://www.piborg.org/sensors-1136/hc-sr04. (Accédé le 09/04/2022).

[4] LastMinuteEngineers. *How HC-SR04 Ultrasonic Sensor Works How to Interface It With Arduino*. URL : https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/. (Accédé le 07/04/2022).

[5] Electrotoile. *L293D pour moteur à courant continu et Arduino*. URL : https://electrotoile.eu/arduino-moteur-DC-shield.php. (Accédé le 09/04/2022).

[6] Wikipédia. *Bipeur — Wikipédia, l'encyclopédie libre*. URL : http://fr.wikipedia.org/w/index.php?title=Bipeur&oldid=178425382. (Accédé le 09/04/2022).

[7] Arduino. *What is Arduino?* URL : https://www.arduino.cc/en/Guide/Introduction. (Accédé le 07/04/2022).

[8] SparkFun Electronics. *Standard Arduino Comparison Guide*. URL : https://www.sparkfun.com/standard_arduino_comparison_guide. (Accédé le 07/04/2022).

[9] Google Maps Platform. *Pricing Plans and API Costs*. URL : https://mapsplatform.google.com/pricing/#places. (Accédé le 11/04/2022).

[10] Google Developers. *Overview | Geocoding API*. URL : https://developers.google.com/maps/documentation/geocoding/overview. (Accédé le 12/04/2022).

[11] Google Developers. *Nearby Search | Places API*. URL : https://developers.google.com/maps/documentation/places/web-service/search-nearby. (Accédé le 12/04/2022).

[12] Google DEVELOPERS. *Place Details | Places API*. URL : https://developers.google.com/maps/documentation/places/web-service/details. (Accédé le 12/04/2022).

[13] Flutter PACKAGE. *flutter_bluetooth_serial*. URL : https://pub.dev/packages/flutter_bluetooth_serial. (Accédé le 12/04/2022).

[14] Arduino REFERENCE. *tone()*. URL : https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/. (Accédé le 13/04/2022).

[15] Axess LAB. *What is a screen reader?* URL : https://axesslab.com/what-is-a-screen-reader/. (Accédé le 13/04/2022).