

FIL ROUGE

PR2

Conception d'un dispositif Bluetooth d'analyse de la marche

Auteurs :

Mathéo GOURDON
Arthur MARIANO
Ianis TRIGUI

Encadrante : Mme Abir REZGUI

11 avril 2021

Remerciements

Nous tenons tout d'abord à remercier l'ESIEE Paris qui nous a permis de mener ce projet avec tous les moyens dont nous avions besoins, ce qui nous a permis de pouvoir suivre un projet enrichissant et passionnant.

Nous souhaitons également remercier notre tutrice Mme Abir REZGUI pour son soutien tout au long du projet, qui a su être là pour nous lorsque nous avions besoin d'informations ou d'explications concernant notre sujet. Grâce à son accompagnement, nous n'avons pas eu de problèmes à nous orienter durant nos recherches.

Enfin, nous souhaitons remercier M. Julien PAGAZANI ainsi que M. Laurent BUES qui nous ont permis de réaliser notre PCB ainsi que notre boîtier, éléments essentiels pour présenter un dispositif utilisable.

Table des matières

1 Contexte	6
1.1 L'équipe du projet	6
1.2 La marche	6
1.2.1 L'Histoire de la marche	6
1.2.2 Cycle de la marche	7
1.2.3 Pourquoi analyser la marche ?	8
1.3 Cahier des charges	8
1.4 Répartition du travail	8
2 Technologies d'acquisition	10
2.1 Capteurs inertIELS : Accéléromètres	10
2.2 Capteurs inertIELS : Gyroscope	13
2.3 Choix des composants	15
2.3.1 Microcontrôleur	15
2.3.2 Centrale inertIELLE	16
3 Conception du prototype	18
3.1 Présentation de la solution technique	18
3.2 Réalisation du schéma électronique et du PCB	19
3.2.1 Réalisation du schéma électronique sous Eagle	20
3.2.2 Réalisation du placement routage	22
3.2.3 Impression du PCB et ajout des composants	24
3.3 Réalisation du boîtier	26
3.4 Connexion sans fil et extraction des données sous Arduino	28
3.4.1 Choix des données à extraire	28
3.4.2 Aperçu des données à extraire	29
3.4.3 Choix du protocole sans fil	30
3.4.4 Contraintes de l'Arduino et de la multi connexion BLE	33
3.5 Réalisation de l'interface sous python	34
3.5.1 Récupération des données	34
3.5.2 Enregistrement des données	35
3.5.3 Affichage courbes	36
3.5.4 Réalisation interface	37
4 Conclusion	40
4.1 Conclusion technique	40
4.1.1 Réponse au cahier des charges	40
4.1.2 Problèmes rencontrés	40
4.1.3 Optimisations possibles	42
4.2 Conclusion personnelle	43

Table des figures

1	Cycle de la marche	7
2	Schéma cycle de la marche	7
3	Diagramme de Gantt	9
4	Capteur inertiel BNO 055	10
5	Accéléromètre de l'ONERA	12
6	Principe d'un accéléromètre capacitif	13
7	Gyroscope	14
8	Photo d'une Nucleo-L432KC et d'une Arduino nano 33 BLE	15
9	Comparatif spécifications des microcontrôleurs	15
10	BNO 055 et LSM9DS1	16
11	Connectivité du système	18
12	Schéma expérimental	19
13	Transmission des données du BNO055	20
14	Schéma fonctionnel du système	20
15	Alimentation	21
16	Partie acquisition de données	21
17	Partie signalétique	22
18	Interface de routage	22
19	Placement des composants	23
20	PCB	24
21	PCB imprimé	24
22	PCB Final	25
23	Conception du boîtier	26
24	Boîtier	27
25	Test du dispositif	27
26	Illustration des angles d'Euler	28
27	Angles d'Euler en sortie du BNO055	29
28	Visualisation des angles d'Euler sur IDE Arduino	30
29	Schéma de fonctionnement du BLE	31
30	Structure de connexion BLE	32
31	Structure de la trame envoyée via le port série	32
32	Comparaison multi connexion et connexion séquentielle BLE	33
33	Frise chronologique de développement réalisé	34
34	Exemple de <i>fichier.txt</i> d'acquisition	35
35	Exemple de <i>fichier.txt</i> ouvert avec Excel	36
36	Affichage courbe avec traitement	36
37	Affichage courbe en temps réel	37
38	Affichage interface terminée	38
39	Affichage interface segments gauche/droite	38
40	Soudures	41
41	Optimisations identifiées	42

1 Contexte

1.1 L'équipe du projet

Notre équipe pour ce projet est composée de trois élèves en cinquième année d'école d'ingénieur en cursus alternant à l'école ESIEE Paris :

- Ianis TRIGUI : chef de projet, alternant à Renault ;
- Mathéo GOURDON : alternant au CEA ;
- Arthur MARIANO : alternant chez Synchrotron SOLEIL.

1.2 La marche

Définition : La marche peut être définie comme un déplacement consistant en une translation de l'ensemble du corps, consécutive à des mouvements de rotations articulaires.

Bouisset et Maton, 1995

Définition : Elle utilise une répétition de séquences des segments corporels pour déplacer le corps vers l'avant en maintenant l'équilibre.

Perry, 1992

1.2.1 L'Histoire de la marche

Tout d'abord, la marche est apparue lorsque la bipédie est arrivée. La bipédie a commencée à apparaître il y a environ deux millions d'années et non pas chez l'Homme, mais chez les animaux. On remarque d'ailleurs aujourd'hui que la plupart des oiseaux ont cette faculté et l'utilise pour se déplacer lorsqu'ils ne volent pas. Ce mode de déplacement a permis la libération des membres avant, ce qui a permis à l'Homme de se redresser et ensuite de se servir de ses mains afin d'attraper des objets, manipuler des outils tout en marchant ou en se tenant debout.

Des études ont commencées à être menées à partir de la Renaissance à travers les travaux de Galilée, Léonard de Vinci ou alors Newton. C'est au XIXème siècle que l'on peut apercevoir les premières descriptions précises du cycle de la marche qui entraîna alors l'estimation de la position du centre de masse humain en position debout et de l'explication des mécaniques d'équilibre entrant en jeu durant la marche en position debout. Toutes ces études ont permises d'acquérir les fondamentaux de la biomécanique de la marche.

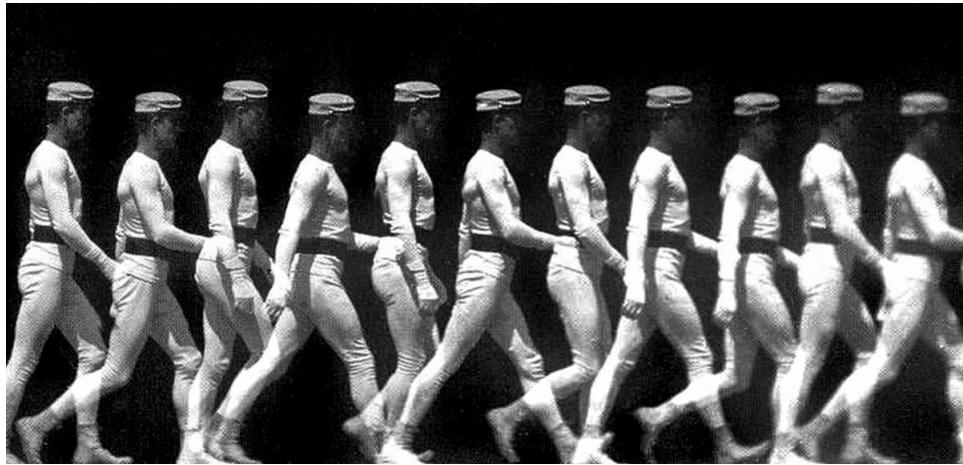


FIGURE 1 – Cycle de la marche

A la suite de l'apparition de l'électronique, des premiers dispositifs d'analyse de marche ont vu le jour, ce qui a permis d'améliorer les modèles mathématiques pour représenter avec précision le cycle de la marche.

1.2.2 Cycle de la marche

Pour un peu mieux comprendre la marche, nous allons étudier son cycle. Le cycle de la marche est séparé en deux étapes principales :

- phase d'appui : pied en contact avec le sol (60%) ;
- phase oscillante : pas de contact du pied avec le sol (40%).

Lors d'un cycle complet, les deux pieds vont toucher ensemble le sol deux fois ce qui représente 20% de notre cycle.

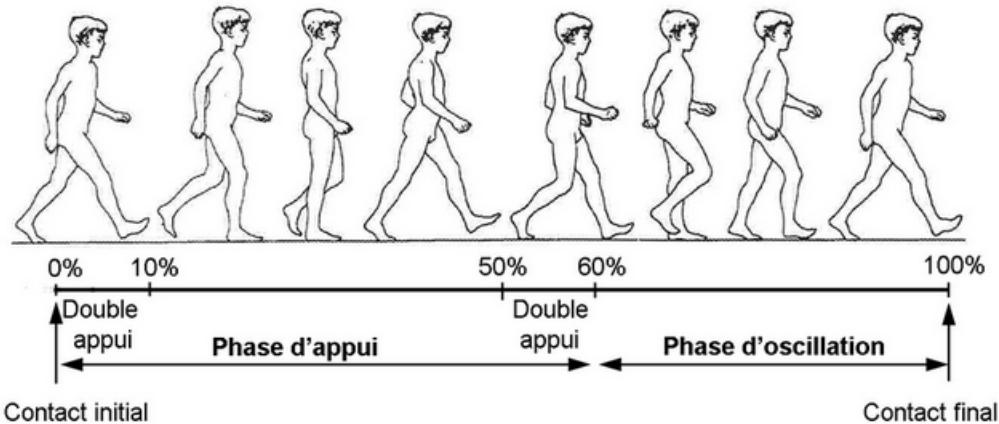


FIGURE 2 – Schéma cycle de la marche

1.2.3 Pourquoi analyser la marche ?

En général, les études qui ont été menées au sujet de l'analyse de la marche avait pour but tout d'abord d'apporter des connaissances nouvelles sur ce sujet, mais aussi pour donner des réponses au sujet du reconditionnement physique des personnes ayant subit des traumatismes d'origine diverse (accidents, maladies...). Ces études peuvent également servir à surveiller le bon fonctionnement des articulations des personnes âgées. Il y a bien entendu encore une multitude d'exemples autres que ceux cités.

Ici, le projet consiste donc à mettre en place un système d'analyse qui permettrait par la suite à identifier, quantifier et comprendre les défauts de marche et ainsi choisir le traitement le plus approprié.

1.3 Cahier des charges

L'objectif global de notre projet est de proposer une solution d'analyse de la marche sans fil et à moindre coût. Sans fil afin de faciliter son utilisation et à moindre coût pour rendre ce genre d'analyses disponibles pour tous.

Pour réaliser ce projet, nous avons souligné 3 grands axes :

- Créer une interconnexion des capteurs en Bluetooth et pouvoir lire les données avec une fréquence d'acquisition de 100 Hz ainsi qu'une précision angulaire de 5° ;
- Réaliser une interface graphique permettant de lire les différentes données reçues des différents capteurs ;
- Réaliser des PCBs pour minimiser le format du dispositif ainsi que réaliser des boîtiers à l'aide d'une imprimante 3D pour rendre le dispositif portable.

1.4 Répartition du travail

Tout d'abord la partie hardware à été séparée de la partie software. Dans la partie hardware nous retrouvons donc :

- La réalisation du PCB ;
- La modélisation 3D du boîtier ainsi que son impression ;

Ces éléments seront à réaliser en fin de projet car ils serviront à la mise en place du dispositif après avoir réalisé les tests du bon fonctionnement global. La partie software est plus importante et correspond au travail principal à réaliser.

Pour la partie software nous aurons :

- L'extraction des données de l'IMU ;
- Connexion Bluetooth des modules Arduino ;
- Connexion au PC ;
- Traitement des données ;
- Interface 3D ;

Afin d'être ordonné dans notre travail, nous avons choisis de réaliser un diagramme de Gantt. Ce diagramme va nous permettre de pouvoir visualiser le travail à faire et de conjecturer sur le temps que va nous prendre chaque tâche. Cette organisation nous permettra de gérer au mieux le projet et de pouvoir terminer les livrables à temps.

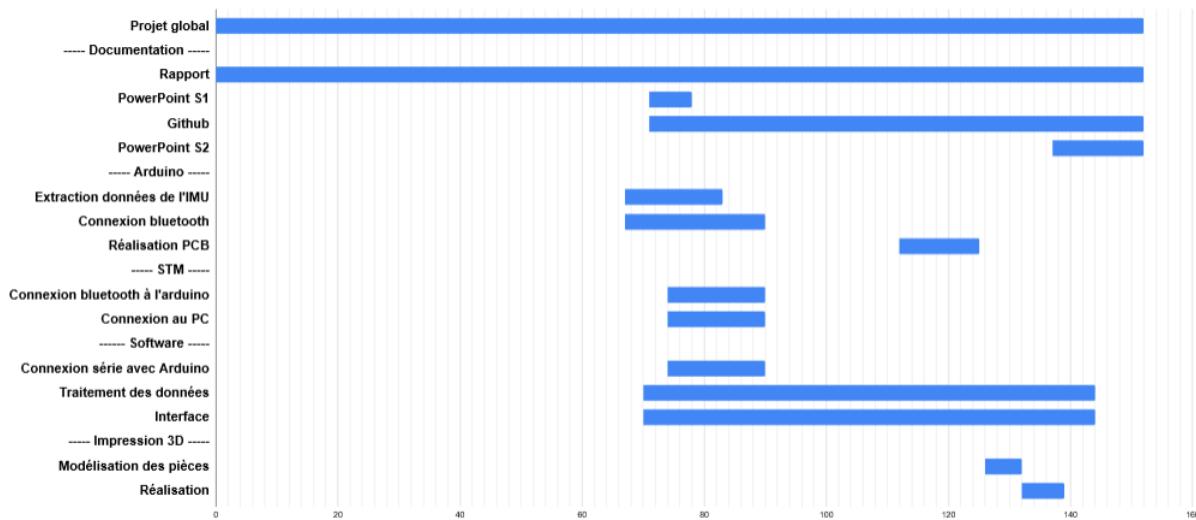


FIGURE 3 – Diagramme de Gantt

La première partie du projet était destinée à de la bibliographie afin de commencer le projet en ayant des connaissances sur ce qu'on allait pouvoir réaliser. Sur la deuxième partie nous avons décidé de commencer par le développement software (Programmation Arduino, Programmation carte STM sur MBED, Connexion Bluetooth avec le PC et interconnexion des cartes Arduino). Ensuite, nous avons réalisé toute la partie hardware qui nous permet de présenter un dispositif fini et présentable.

2 Technologies d'acquisition

Il existe plusieurs manières de mesurer les données relatives aux mouvement du corps, mais dans notre cas nous avons voulu utiliser des systèmes inertiel. Ces systèmes sont jugés précis, peu cher et facile à utiliser et ont l'avantage d'avoir un format miniaturisé. De plus, ils sont utilisables avec des systèmes d'exploitation open source, ce qui permet de les utiliser à moindre coût et en les couplant avec des microcontrôleurs tel que Arduino, il est très facile de les rendre portables et donc de les intégrer directement à notre dispositif.

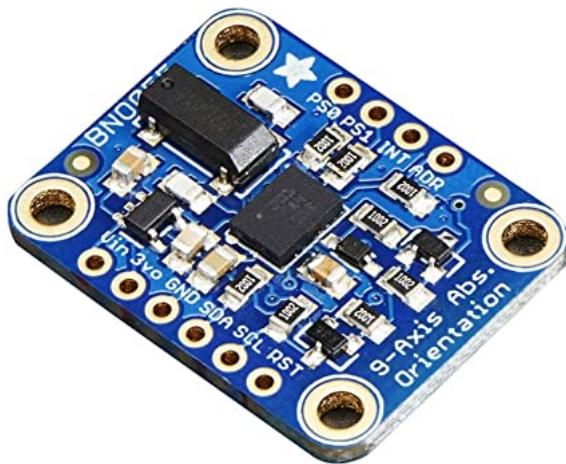


FIGURE 4 – Capteur inertiel BNO 055

2.1 Capteurs inertiel : Accéléromètres

Dans un premier temps on retrouve dans la famille des capteurs inertielles les accéléromètres. Depuis les années 90, des accéléromètres résonnantes sont développés pour le domaine civil et militaire. Ils sont utilisés pour de multiples applications, comme par exemple le contrôle d'altitude, de vitesse, et dans notre cas pour l'acquisition des données de mouvements. Leur haute précision est très importante dans le domaine spatial et aéronautique et doit être de 10^{-5} à 10^{-6} pour être sûr du bon fonctionnement de l'application.

On voit que la miniaturisation présente des avantages importants pour le domaine d'utilisation de l'accéléromètre. Un accéléromètre est un capteur qui, composé d'une masse mobile fixée sur un support permet de mesurer l'accélération linéaire du support en mesurant le déplacement relatif de la masse par rapport à son support. Il est caractérisé par plusieurs paramètres dont :

- La plage de fonctionnement : Valeurs extrêmes pouvant être mesurées ;
- La sensibilité : Variation du signal de sortie par rapport à la variation du signal d'entrée ;

- La fréquence de résonance : fréquence caractéristique d'un système dont la variation permet de mesurer l'accélération ;
- La résolution : Plus petite variation d'accélération mesurable ;
- La bande passante : intervalle de fréquences dans lequel l'affaiblissement du signal est inférieur à une valeur spécifiée ;
- Le facteur qualité : rapport de la fréquence propre à la bande passante. Il est sans unité et mesure le taux d'amortissement d'une oscillation qui est essentiellement dû aux frottements de l'air pendant le mouvement ;
- Le facteur d'échelle : la variation de la fréquence de résonance est proportionnelle à l'accélération appliquée. Le facteur d'échelle est le coefficient de proportionnalité correspondant ;
- La Sensibilité aux accélérations transversales : Variation du signal de sortie dû aux forces engendrées à la poutre dans un autre axe que celui pour lequel il est conçu. Nous expliquons ce principe d'axes plus tard ;
- Le seuil : Différence entre la valeur réelle de la mesure et celle obtenue à partir de la réponse d'un capteur pour la borne inférieure de l'étendue de mesure ;
- Le choc maximum toléré : Représente le niveau de choc que le capteur est capable de supporter avant de montrer un disfonctionnement ;
- Le bruit : Signaux parasites qui viennent se superposer au signal dit utile.

Ils peuvent être fabriqués en différentes matières, essentiellement en silicium ou en quartz. Le quartz présente, de par sa stabilité thermique, un avantage important vis-à-vis du silicium pour la conception d'accéléromètres à lames vibrantes. Le contrôle de la déviation due à la température est très important car il permet d'obtenir une bonne précision.

Les accéléromètres sont regroupés en plusieurs familles selon leur fonctionnement. Les trois principales sont les capacitifs, les vibrants et les piézorésistifs.

Pour ces trois familles une des clés de leur intégration dans les systèmes embarqués est leur SWaP (Size Weight and Power). Optimiser le SWaP d'un capteur consiste à réduire la taille, le poids et la consommation d'un capteur. Cette partie est difficile car il faut réussir à réduire les dimensions ainsi que la puissance consommée sans impacter le fonctionnement du capteur. L'étude des composants MEMS est alors un sujet qui va beaucoup nous intéresser.

Une manière très commune de mesurer l'accélération consiste à exploiter une variation de fréquence de résonance d'un système. On retrouve généralement sur ce genre de structure une poutre dont la contrainte axiale varie sous l'effet d'une accélération. Ce phénomène provoque une variation de la fréquence de résonance en flexion de la poutre. Le fonctionnement est le même que celui d'une corde de guitare, l'accélération va donc être mesurée grâce à la variation de fréquence.

Plusieurs architectures d'accéléromètres à poutres vibrantes existent. On peut voir Figure 4 un exemple d'accéléromètre en quartz réalisé par l'ONERA et très connu.

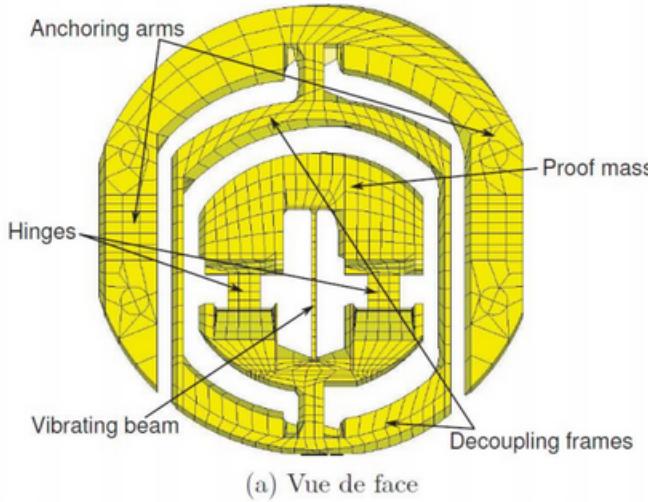


FIGURE 5 – Accéléromètre de l'ONERA

Pour que ce système fonctionne, une masse va être reliée à la poutre. Lorsqu'une accélération va avoir lieu, la masse va se déplacer et va modifier la contrainte appliquée sur le système vibrant. La fréquence de résonance des poutres va donc varier selon un facteur d'échelle. L'équation suivante exprime la variation de fréquence en fonction de l'accélération :

$$F = F_0 + K \times a \quad (1)$$

Où a représente la projection de l'accélération sur l'axe sensible de l'accéléromètre. Si nous connaissons la fréquence de résonance au repos F_0 ainsi que le facteur d'échelle K , nous pouvons alors déterminer la valeur de l'accélération appliquée sur l'axe sensible.

Ces accéléromètres doivent donc être utilisés par paire. En effet, ces deux systèmes résonants montés tête-bêche répondent de façon opposée à une accélération. Si les deux systèmes vibrants disposent de la même fréquence de résonance (F_0 identique) et du même facteur d'échelle (K identiques), alors mesurer l'écart de fréquence entre les deux systèmes résonants revient à mesurer l'accélération.

L'équation suivante exprime l'intérêt d'une telle mesure dans un cas idéal :

$$F_1 = F_0 + K \times a \quad (2)$$

$$F_2 = F_0 - K \times a \quad (3)$$

$$\Delta F = F_1 - F_2 \quad (4)$$

$$\Delta F = 2K \times a \quad (5)$$

La simplicité pour récupérer le signal de sortie est également un atout pour ce genre de système. En effet, les accéléromètres à lames vibrantes permettent de récupérer une sortie

quasiment numérisée. Comme c'est à une fréquence que nous avons à faire, le système électronique numérique de traitement est plus simple et peut donc fournir plus rapidement les valeurs en sortie que l'unité de calcul pourra alors exploiter.

Dans la famille des capteurs inertIELS nous retrouvons ensuite les capteurs capacitifs. Ils sont souvent modélisés à l'aide d'électrodes parallèles. Leur fonctionnement est simple, nous allons nous servir de capacités variables pour nous donner la valeur d'un déplacement, d'une accélération ou de toutes autres mesures. Les grandeurs qui permettent de connaître les valeurs souhaitées sont :

- La permittivité du vide ε_0 ;
- La permittivité relative du matériau entre les électrodes ε_r ;
- La surface effective des électrodes face à face S ;
- Le gap entre les électrodes h .

$$C = \frac{\varepsilon_0 \times \varepsilon_r \times S}{h} \quad (6)$$

Prenons l'exemple d'un accéléromètre (voir Figure 5). Lors d'une accélération, la masse mobile où est fixée une des électrodes va se mettre en mouvement. L'écart entre l'électrode présente sur la masse mobile et l'électrode fixe va être modifié. Cela va engendrer une variation de capacité qui va nous permettre de déduire une accélération.

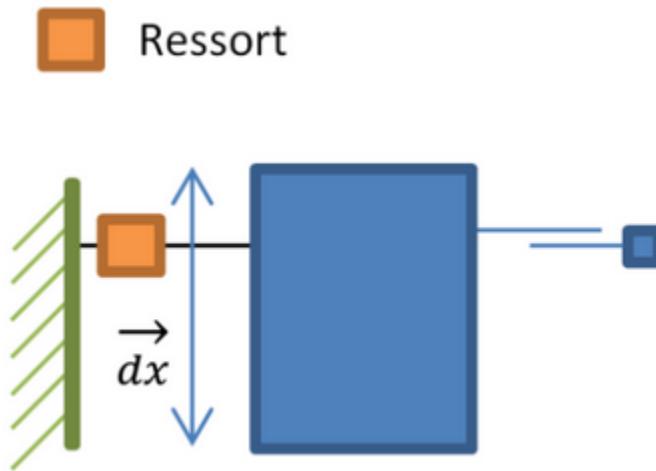


FIGURE 6 – Principe d'un accéléromètre capacitif

Ce type de capteur est souvent utilisé par leur simplicité d'utilisation.

2.2 Capteurs inertIELS : Gyroscope

Les gyroscopes sont des capteurs inertIELS utilisés pour détecter des postions angulaires. Son fonctionnement est basé sur la conservation du moment angulaire, c'est le moment de la

quantité de mouvement \vec{p} par rapport au point O, c'est-à-dire le produit vectoriel :

$$\vec{L}_0 = \overrightarrow{OM} \wedge \vec{p} \quad (7)$$

L'essentiel du dispositif est une lourde roue dont la masse est reportée à la périphérie dénommée tore tournant à grande vitesse sur son axe.

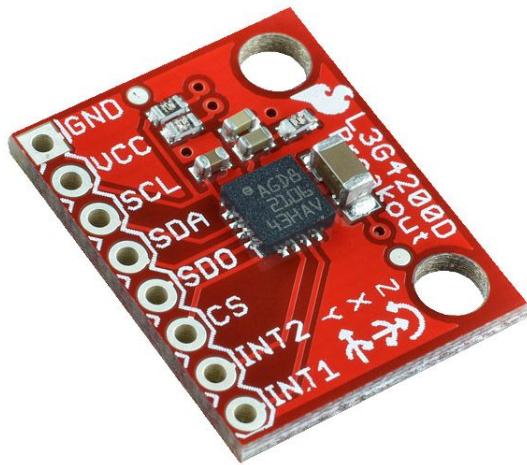


FIGURE 7 – Gyroscope

Le comportement du gyroscope peut être traduit par l'équation :

$$\vec{\tau} = \frac{d\vec{L}}{dt} \quad (8)$$

Dans cette expression, $\vec{\tau}$ désigne le moment ou le couple sur le gyroscope et \vec{L} désigne le moment cinétique.

2.3 Choix des composants

Dans cette partie, nous allons décrire les différents composants que nous avons choisis par rapport à ce que nous pouvions trouver sur le marché.

2.3.1 Microcontrôleur

Le premier composant que nous devions trouver est notre microcontrôleur. Ce microcontrôleur doit être petit et portable car il sera accroché à des jambes. Il doit aussi avoir une fréquence d'acquisition assez élevée (au moins 10 fois supérieure à 100Hz), il doit pouvoir communiquer en sans fil et doit avoir un coût assez faible.

Notre choix s'est posé sur deux fournisseurs de microcontrôleur. Le premier est Arduino et le second STM.

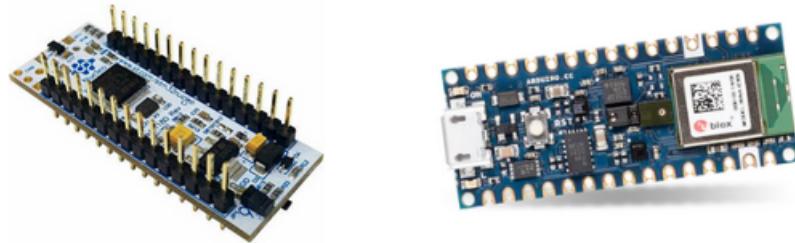


FIGURE 8 – Photo d'une Nucleo-L432KC et d'une Arduino nano 33 BLE

Nos deux micro-contrôleurs ci-dessus ont une taille relativement portable et répond dans un premier temps à notre exigence de grandeur.

	NUCLEO-L432KC	Arduino Nano 33 BLE Sense	Arduino Nano 33 BLE
Flash memory	246 kB	1 MB	1 MB
Embedded IMU	NO	YES	YES
Embedded Bluetooth	NO	YES	YES
Clock speed	80 MHz	64 MHz	64 MHz
Price	Low $\simeq 10\text{€}$	High $\simeq 27\text{€}$	Medium $\simeq 17.50\text{€}$

FIGURE 9 – Comparatif spécifications des microcontrôleurs

Nous avons retenu 3 microcontrôleurs à comparer :

- la NUCLEO-L432KC ;
- l'Arduino Nano 33 BLE Sense ;
- l'Arduino Nano 33 BLE.

La première carte est la **NUCLEO-L432KC**. Cette carte à l'avantage d'avoir un prix relativement faible car nous ne devons pas oublier que nous allons avoir 7 microcontrôleurs interconnectés entre eux. Cependant, malgré sa fréquence d'acquisition et sa mémoire flash suffisante, elle ne possède pas de module sans fil ni d'IMU. Cela implique des coûts et une taille de la solution par conséquents supérieurs à ce que nous souhaitons.

Notre choix va donc se porter sur une des deux Arduino.

Nos deux arduino (Nano 33 BLE Sense et Nano 33 BLE) ont les mêmes caractéristiques pour les besoins que nous recherchons. L'arduino Nano 33 BLE Sense cependant embarque des capteurs de température et de pression, mais qui nous sont inutiles.

Nous retiendrons comme microcontrôleur l'**Arduino Nano 33 BLE** car elle est 10 € moins chère que son homologue.

2.3.2 Centrale inertielle

Afin de faire l'acquisition des données inertielles nous allons nous servir d'un IMU. Plusieurs choix s'offrent à nous :

- BNO 055 ;
- LSM9DS1.

Chacune de ces centrales inertielles présente des avantages et des inconvénients que nous comparerons afin de se fixer sur la technologie à utiliser. Afin de répondre à notre cahier des charges, nous constatons que nous avons besoin d'une précision angulaire de 5° ainsi qu'une fréquence d'acquisition minimale de 100 Hz. Afin de rendre notre dispositif portable facilement, un élément à prendre en compte également est la dimension de l'IMU. Entre les deux systèmes d'acquisition, on remarque que le BNO 055 est un composant à ajouter au système alors que la LSM9DS1 est déjà embarquée sur la carte Arduino nano 33 BLE.

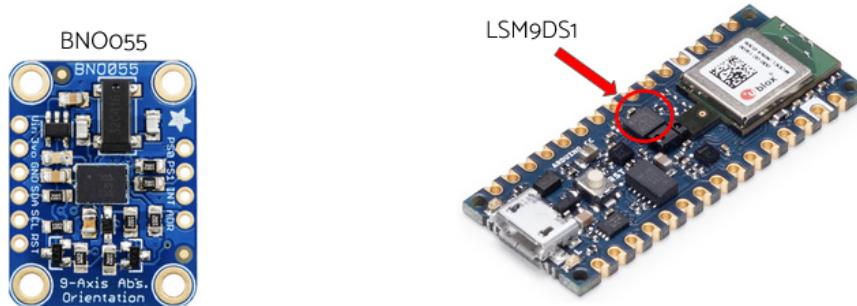


FIGURE 10 – BNO 055 et LSM9DS1

Il serait alors logique de se porter vers le choix de la LSM9DS1, mais nous allons par la suite comparer les différentes spécifications pour s'assurer que la BNO 055 ne serait pas un choix plus judicieux. Dans le tableaux ci-dessous, nous pouvons comparer succinctement les différentes technologies.

	BNO055	LSM9DS1
Acceleration data range	± 2 to ± 16 g	± 2 to ± 16 g
Magnetometer data range	± 1200 to ± 2500 μ T	± 200 to ± 1600 μ T
Gyroscope data range	125 to 2000 °/s	245 to 2000 °/s
Data fusion (Euler angles, Quaternion)	YES	NO
Output data rate	Acc : 100 Hz Mag : 20 Hz Gyro : 100 Hz Fusion data : 100 Hz	Max 80 Hz

Nous remarquons ci-dessus que les deux IMUs présentent globalement les mêmes caractéristiques. Deux paramètres vont ici nous intéresser. Le premier est la vitesse d'envoie des données. En effet, on remarque que pour valider le cahier des charges, une fréquence d'acquisition de 100 Hz est demandée. Cette vitesse est validée par le BNO 055 mais pas par la LSM9DS1. Il serait possible de faire une concession mais il serait préférable d'avoir cette vitesse minimale de 100 Hz. La BNO055 présente également l'avantage de proposer une technologie de data fusion. Cette caractéristique est très importante car elle permet de traiter les données bien plus facilement.

Pour les raisons citées ci-dessus nous avons décidé de nous porter sur le choix de la BNO055.

3 Conception du prototype

3.1 Présentation de la solution technique

Pour répondre à la problématique d'analyser le plus fidèlement possible la marche, nous avons décidé de fixer des capteurs inertIELS sur chaque segment de la jambe. Un premier au niveau de la cuisse, un second au niveau du tibia, et enfin un dernier au niveau du pied. L'objectif ici est de pouvoir fournir des données pour chaque segment de la jambe.

Notre problématique exige que le système soit facile à porter et puisse gêner le moins possible l'individu qui porte le dispositif. C'est pourquoi tous les capteurs seront connectés entre-eux grâce à une communication BLE (Bluetooth Low Energy). Tandis que 6 dispositifs auront comme fonction de faire l'acquisition des données inertielles, la septième carte va servir à la fois de collecte d'informations inertielles mais également de centralisation des données collectées par les 6 autres cartes. Cette septième carte pourra ensuite transmettre par l'intermédiaire d'un liaison USB les données collectées.

Un axe d'amélioration possible est de proposer une communication entre le PC et la septième carte non pas en liaison USB, mais avec une connexion BLE.

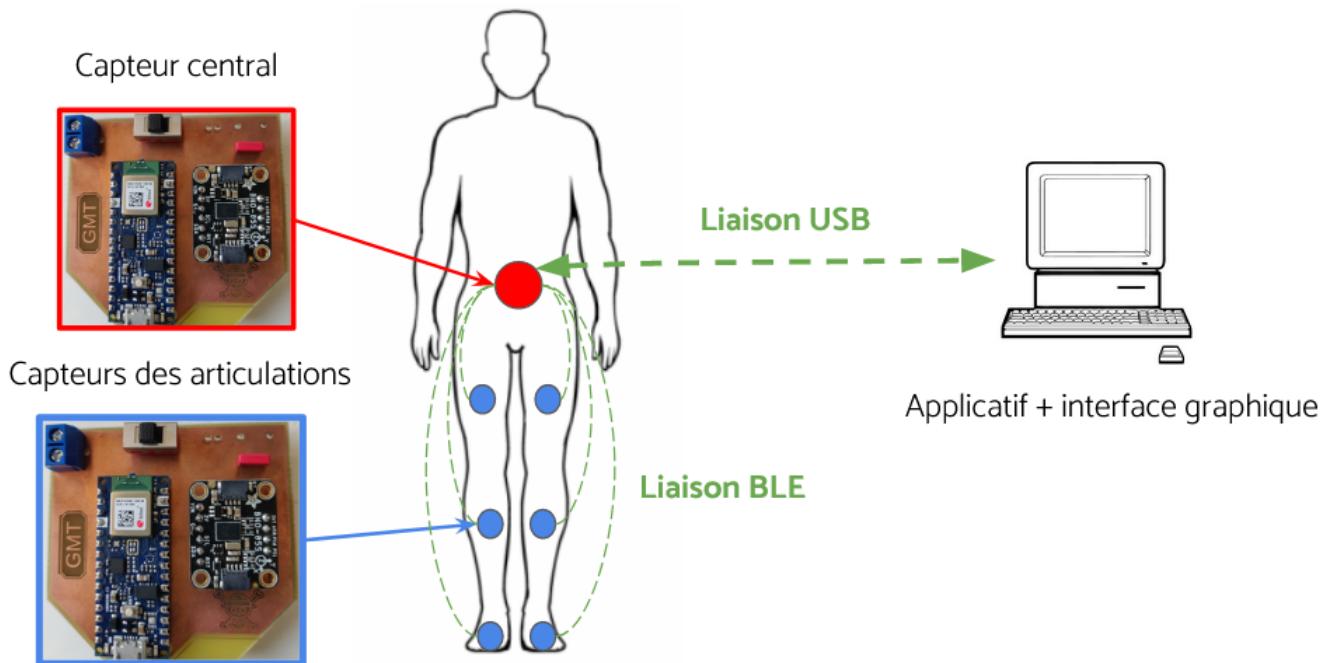


FIGURE 11 – Connectivité du système

3.2 Réalisation du schéma électronique et du PCB

Dans un premier temps, le but est de réaliser un montage expérimental entre une carte Arduino Nano 33 BLE ainsi qu'une centrale inertuelle BNO055 qui permettrait de valider le bon fonctionnement de notre montage. Pour cela, le schéma réalisé a été le suivant.

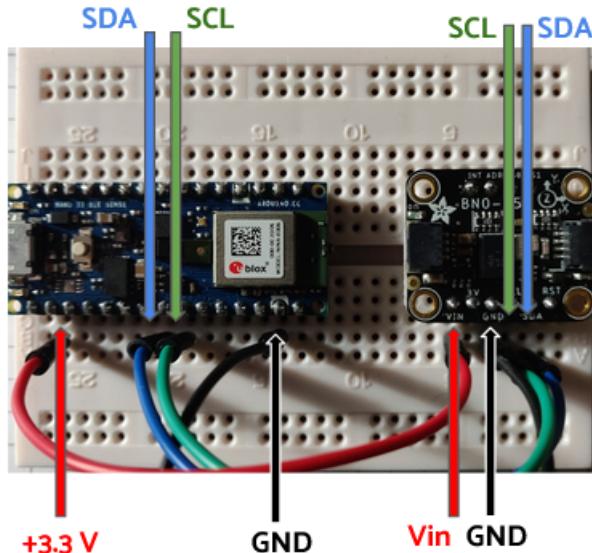
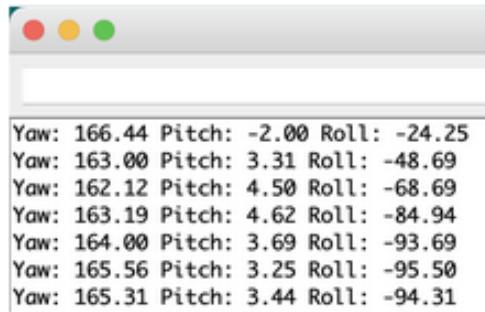


FIGURE 12 – Schéma expérimental

On remarque donc sur la Figure 12 que nous nous servons d'une breadboard afin de venir fixer nos différents composants et de pouvoir tester son fonctionnement. Le montage est assez simple, dans un premier temps nous alimentons notre carte Arduino avec une alimentation 9V et pour alimenter le BNO055 nous connectons la sortie 3.3V de la carte Arduino à la broche *Vin* de la centrale inertuelle. Les deux composants sont bien entendu reliés à la masse et nous avons fait le choix pour communiquer d'utiliser le bus I2C de ces deux cartes. Pour les faire communiquer entre-elles les broches SDA et SCL ont donc été reliées aux broches associées sur chaque composant.

Grâce à une connexion série avec le PC nous avons pu voir si la communication entre l'Arduino et le PC fonctionnait bien et également si le BNO055 transmettait des données. Tout d'abord nous avons vu que lors de la connexion, les deux composants étaient bien alimentés. Ensuite sur le moniteur de l'IDE nous avons pu vérifier que la centrale inertuelle transmettait bien des données.

Nous pouvons voir sur la Figure 13 que les données étaient bien transmises et nous allions pouvoir passer à l'élaboration d'un schéma électrique sous Eagle pour en finalité réaliser un PCB de notre dispositif.



```

Yaw: 166.44 Pitch: -2.00 Roll: -24.25
Yaw: 163.00 Pitch: 3.31 Roll: -48.69
Yaw: 162.12 Pitch: 4.50 Roll: -68.69
Yaw: 163.19 Pitch: 4.62 Roll: -84.94
Yaw: 164.00 Pitch: 3.69 Roll: -93.69
Yaw: 165.56 Pitch: 3.25 Roll: -95.50
Yaw: 165.31 Pitch: 3.44 Roll: -94.31

```

FIGURE 13 – Transmission des données du BNO055

3.2.1 Réalisation du schéma électronique sous Eagle

Dans un premier temps nous devons définir quelles fonctions nous voulons réaliser sur notre carte. Nous avons donc besoin d'une alimentation pour alimenter la carte Arduino, d'une partie composant comprenant la carte Arduino ainsi que le BNO055 qui nous servira d'acquisition des données inertielles et enfin d'une partie signalétique qui nous informera sur l'état du système.

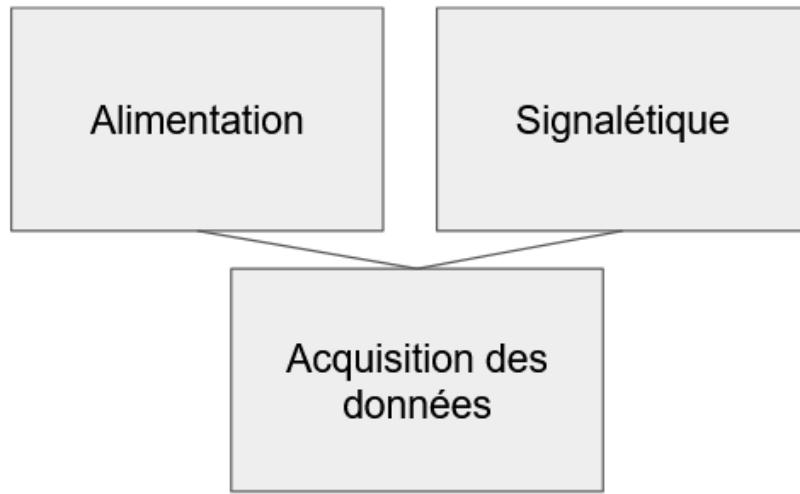


FIGURE 14 – Schéma fonctionnel du système

Ensuite nous allons parler de la partie alimentation. Nous avons fait le choix d'utiliser une alimentation 9V qui sera reliée à un interrupteur qui permettra de mettre ou non notre dispositif sous tension. Afin de fiabiliser notre système ainsi que pour éviter des montées ou des chutes de tension trop importantes nous avons ajouté un condensateur.

Nous pouvons voir que l'interrupteur est un commutateur 3 états (ON/OFF/ON). C'est pourquoi les deux pins ON ont été reliés entre eux afin de mettre le système sous tension peu importe la position ON sélectionnée.

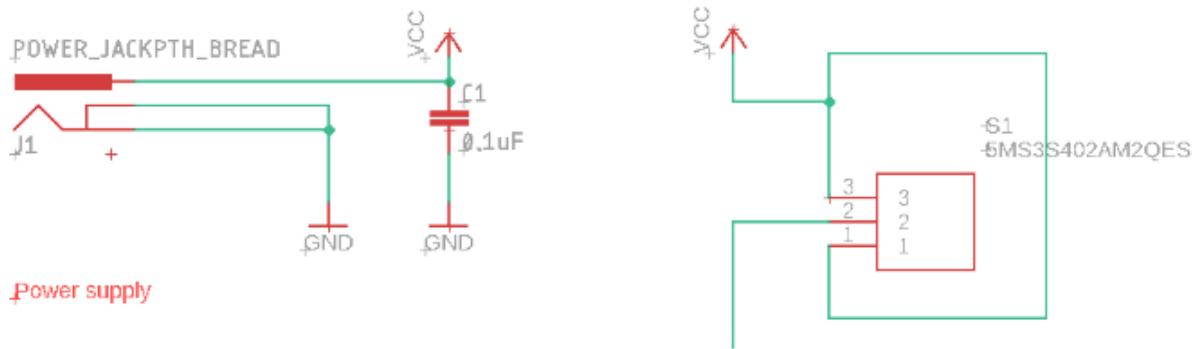


FIGURE 15 – Alimentation

Ensuite pour la partie acquisition de données, nous avons réalisé le même montage que vu précédemment. L'alimentation venant de l'interrupteur va être reliée au pin *Vin* de l'Arduino. C'est celle-ci qui par l'intermédiaire de sa sortie 3.3V va permettre d'alimenter le reste du dispositif avec une tension plus adaptée aux autres composants. Pour l'échange de données, nous nous servons du bus I2C et nous connectons donc les pins SCL et SDA de la carte Arduino à ceux de la BNO055.

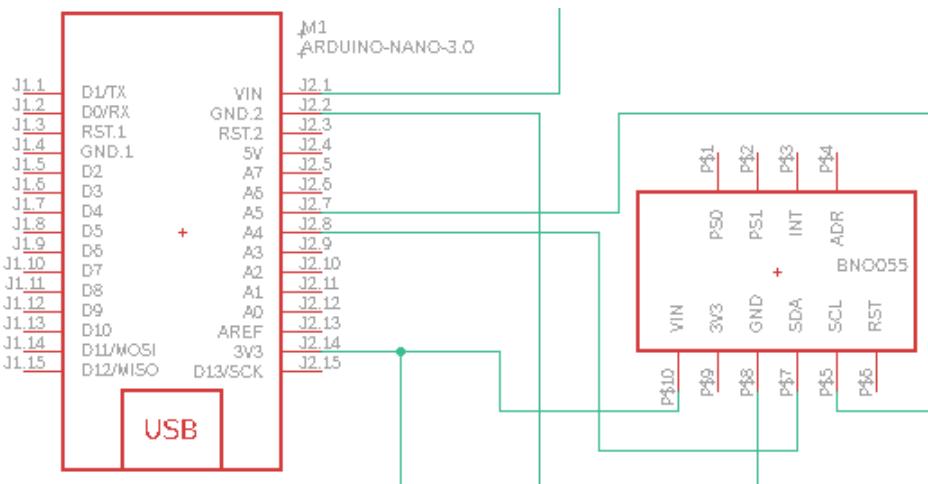


FIGURE 16 – Partie acquisition de données

Et pour finir, nous avons ajouté une partie signalétique qui va nous permettre de montrer lorsque le système est alimenté. Pour cela nous avons utilisé une led qui sera protégée par une résistance. Cette led est également alimentée par le pin 3.3V de la carte Arduino.

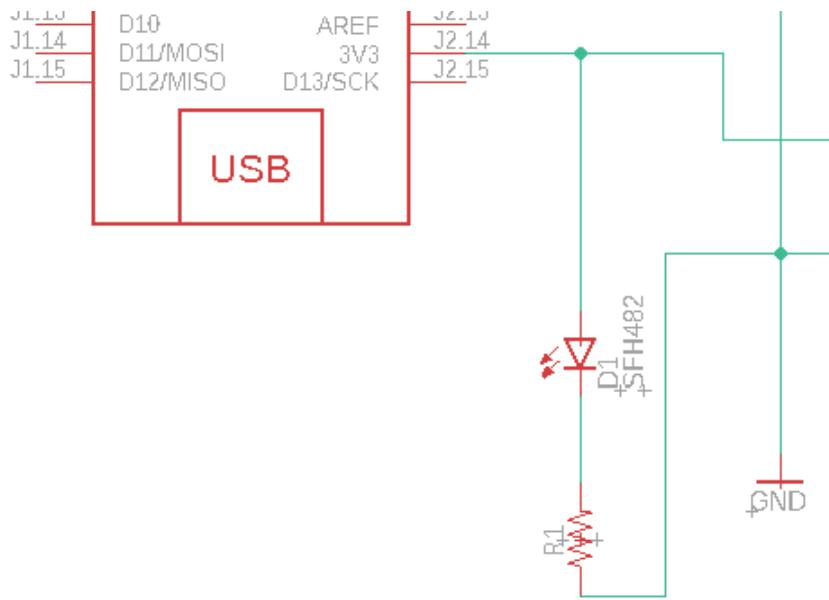


FIGURE 17 – Partie signalétique

Après validation de ce schéma nous pouvons passer au placement routage du PCB, toujours sous l'IDE Eagle.

3.2.2 Réalisation du placement routage

Après validation du schéma, l'IDE Eagle permet de passer directement au placement routage à partir du schéma réalisé précédemment. Nous avons donc l'interface suivante avec chaque composant relié à une empreinte définie par les dimensions réelles.

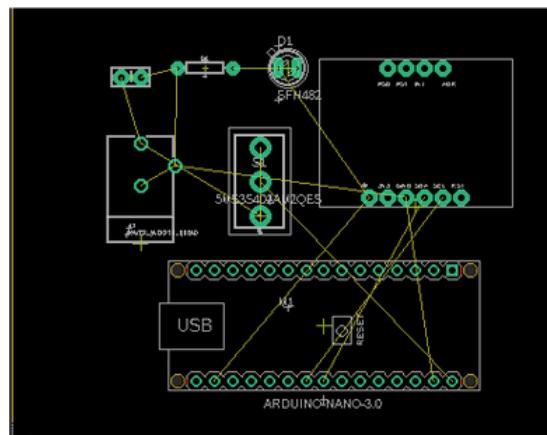


FIGURE 18 – Interface de routage

Nous pouvons donc voir sur la figure 22 ci-dessus que chaque composant est représenté :

- L'alimentation ;
- Les composants passifs (résistances...) ;
- La carte Arduino Nano 33 BLE ;
- La centrale inertielle BNO055.

On remarque également des liaisons jaunes entre chaque composant. Ces liaisons correspondent aux connexions que nous allons faire entre chaque pin. Le but sera ici d'essayer de placer les composants et de faire passer les pistes de façon à ce que la carte soit la plus petite possible et également de façon à ce qu'il n'y ait pas de via (trou à travers la carte pour faire passer des pistes sur l'autre face) ni de croisement entre les pistes.

Après placement des composants et placement des différentes pistes nous avons à ce schéma.

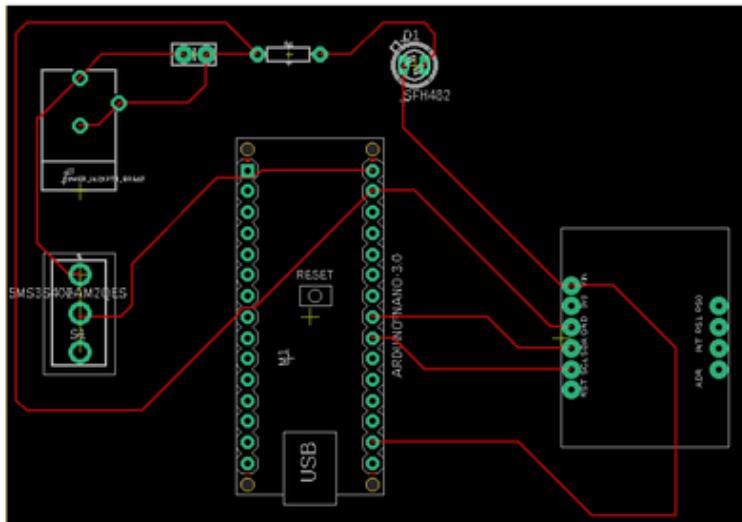


FIGURE 19 – Placement des composants

Comme on peut le remarquer, aucune piste ne se croise mais nous pouvons encore gagner de la place. Il est important de vérifier les dimensions de chaque empreinte pour ne pas être surpris après réalisation des cartes avec des composants qui ne rentreraient pas dans les empreintes prévues à cette effet.

Une fois chaque dimension vérifiée nous pouvons réduire l'espace entre chacun de nos composants. Il faut savoir que les schémas vus précédemment ne sont pas ceux que nous avons utilisés pour le dispositif. En effet, il s'agit d'un nouveau schéma réalisé car celui utilisé présentait des problèmes (dimensions d'empreintes, liaisons des pins, etc). Nous soulignons donc qu'il faut bien vérifier chaque dimension et chaque liaison avant de faire réaliser le PCB.

Nous pouvons observer sur la prochaine Figure le PCB que nous avions réalisé après optimi-

sation de la place.

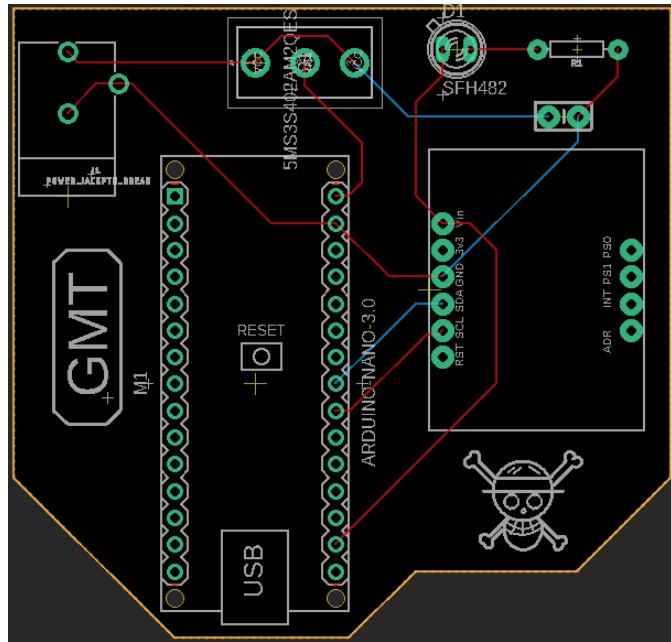


FIGURE 20 – PCB

3.2.3 Impression du PCB et ajout des composants

Une fois après avoir validé ce prototype, nous allons donc faire une demande afin de faire imprimer les 7 cartes dont nous avions besoin. Une fois réalisé, nous obtenons le prototype suivant :

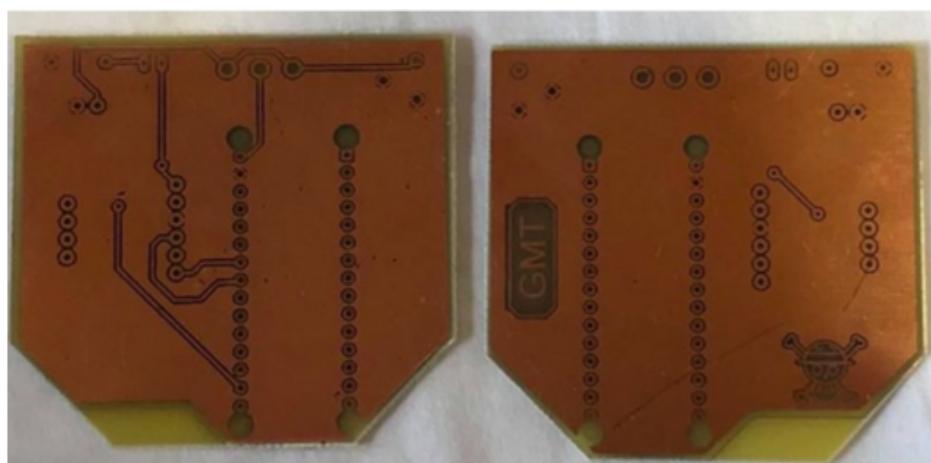


FIGURE 21 – PCB imprimé

Pour les raisons citées précédemment le PCB imprimé ne correspond pas précisément à sa conception sous Eagle.

La prochaine étape va être de percer les différentes pastilles du PCB afin de pouvoir ensuite insérer les pins de chacun de nos composants. Pour ce faire nous nous sommes servis de perceuse prévues à cette effet avec des forets aux bonnes dimensions. Une fois les différents composants disposés sur le PCB nous pouvons les souder pour qu'ils ne bougent plus. Une fois soudés, nous avons notre premier prototype de réalisé.



FIGURE 22 – PCB Final

On remarque ici que la led ainsi que sa résistance associée ne sont pas présents. Pour un premier prototype nous voulions tout d'abord valider son bon fonctionnement. Au premier essai nous nous sommes rendu compte que la carte n'était pas alimentée. Nous avons donc dû analyser le problème et nous avons trouvé une soudure qui créait un court-circuit sur le pin *Vin* de l'Arduino. Ensuite la carte Arduino était bien alimenté mais des problèmes de court-circuit empêchaient encore le fonctionnement de la centrale inertuelle. Et enfin nous avions des problèmes vis-à-vis de la transmission de données. En effet, tout le dispositif était bien alimenté mais nous ne pouvions plus lire de données sur le traceur série de l'IDE Arduino. Nous nous sommes rendu compte que les pins *SDA* et *SCL* n'étaient pas bien raccordés.

Une fois tous ces problèmes résolus le prototype fonctionnait bien et permettait à nouveau de lire les données de la centrale inertuelle sur le traceur série.

3.3 Réalisation du boîtier

Pour le boîtier nous avons choisis d'utiliser l'IDE en ligne SketchUp qui est gratuit. Pour réaliser ce boîtier nous avons tout d'abord relevé les différentes dimensions des composants qui seront présents dans celui-ci. Les dimensions à prévoir sont donc :

- La pile 9V : 26 x 48 x 17mm ;
- Le PCB avec les composants : 60 x 62 x 150mm.

Une fois que nous avons toutes les dimensions, nous pouvons passer à la conception du boîtier. Tout d'abord nous avons réalisé un boîtier avec comme dimensions intérieures 67 x 67mm pour être sûr de pouvoir ajouter et retirer facilement notre dispositif. Ensuite nous avons rajouté au fond du boîtier des cales pour éviter que la pile 9V ne bouge trop dans le boîtier et entraîne des désagréments lors de l'utilisation du dispositif. Pour le couvercle, nous avons choisi d'utiliser une liaison pivot sur chaque extrémité du boîtier.

Après conception, nous obtenons ce prototype.

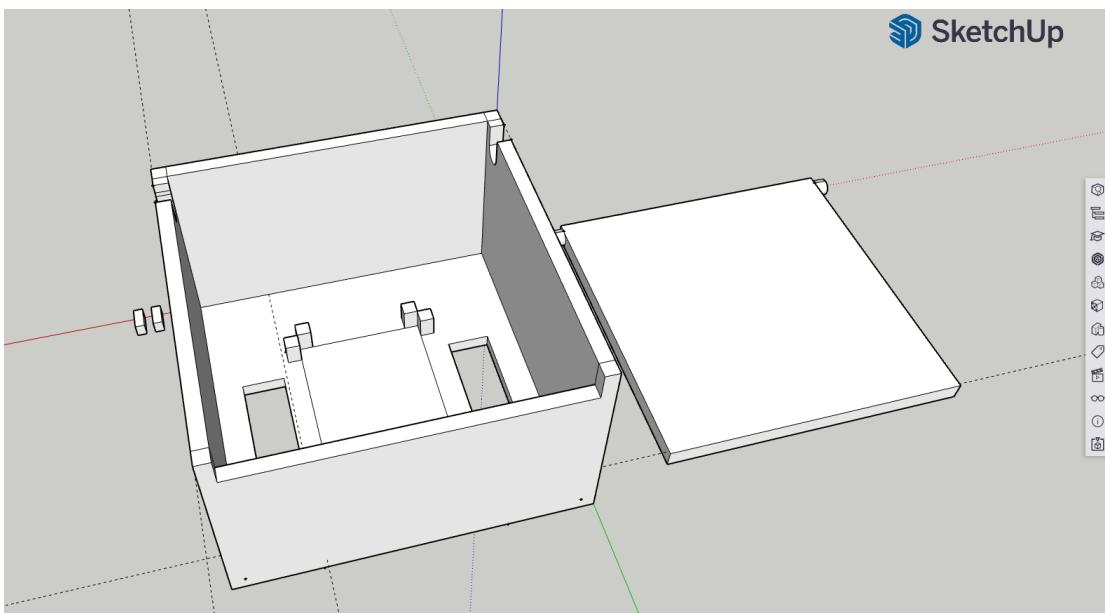


FIGURE 23 – Conception du boîtier

Une fois le prototype sous SketchUp terminé, nous l'avons donc imprimé grâce à une imprimante 3D. On peut voir le résultat de l'impression ci-dessous.

On observe également que des fentes ont préalablement été placées au fond du boîtier pour permettre de faire passer des lanières scratch afin de pouvoir attacher chaque prototype sur les segments de la jambe. Après vérification, nous avons pu valider le prototype du boîtier et ses dimensions car tous les éléments du dispositif peuvent rentrer dedans.



FIGURE 24 – Boîtier

Une fois le PCB et le boîtier réalisés nous avons pu tester si l'ensemble permettait bien une utilisation simple et que les fixations choisies permettaient de bien fixer le dispositif sur les jambes. Pour cela nous avons utilisé 3 dispositifs que nous avons par la suite fixés. Nous pouvons voir ci-dessous le résultat du test.



FIGURE 25 – Test du dispositif

Ce test nous a permis de valider l'ensemble de chaque réalisation.

3.4 Connexion sans fil et extraction des données sous Arduino



La partie qui sera réalisée sous arduino aura pour rôle d'extraire les données envoyées par la centrale inertuelle, puis d'établir une connexion sans fil avec l'ensemble des capteurs afin de pouvoir transmettre l'intégralité des données des mouvements de jambes au PC.

Dans cette partie nous mettrons en interconnexion l'ensemble des 7 Arduino Nano 33 BLE ainsi qu'avec les capteurs inertIELS que sont les BNO055.

3.4.1 Choix des données à extraire

Une fois l'Arduino et le BNO055 connectés nous allons désormais choisir les données à extraire de la centrale inertuelle. Le BNO055 propose un large choix de données à extraire, car étant une centrale inertuelle à 9 degrés de liberté doté d'un magnétomètre, gyroscope et accéléromètre le choix est large. Néanmoins l'objectif derrière l'extraction des données du BNO055 est de choisir des outputs qui puissent nous délivrer l'orientation de notre capteur dans l'espace.

Heureusement, le BNO055 possède un algorithme des fusions des data issues du magnétomètre, gyroscope et accéléromètre. Cet algorithme nous sort des datas qui nous donnent l'orientation absolue du capteur.

Il existe deux types de data fusion en sortie du BNO055 :

- Les quaternions ; ;
- Les angles d'Euler.

Par souci de compréhension nous avons choisi d'exploiter les angles d'Euler. Les angles d'Euler se décomposent en trois coordonnées :

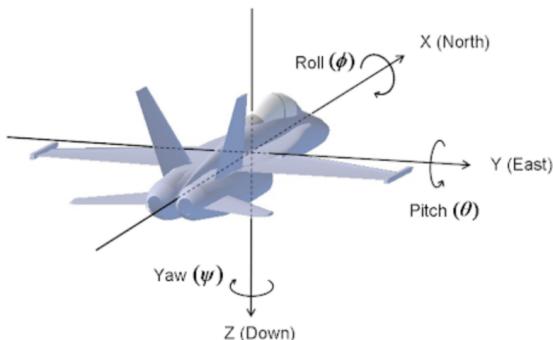


FIGURE 26 – Illustration des angles d'Euler

- Le Yaw (lacet) nous donne le mouvement de rotation horizontal d'un mobile autour de son axe vertical
- Le Pitch (tangage) nous donne le mouvement de rotation autour de l'axe transversal d'un objet en mouvement
- Le Roll (roulis) est un mouvement de rotation d'un mobile autour de son axe longitudinal

Voici ci-dessous le format de chacune de ces données en sortie du BNO055 :

Rotation angle	Range (Android format)	Range (Windows format)
Pitch	+180° to -180° (turning clockwise decreases values)	-180° to +180° (turning clockwise increases values)
Roll	-90° to +90° (increasing with increasing inclination)	
Heading / Yaw	0° to 360° (turning clockwise increases values)	

FIGURE 27 – Angles d'Euler en sortie du BNO055

Le BNO055 propose une fréquence d'acquisition de 100 Hz pour l'envoi des angles d'Euler, ce qui est cohérent avec notre cahier des charges.

3.4.2 Aperçu des données à extraire

Par la suite, toujours sous l'IDE Arduino nous extrayons les angles d'Euler du BNO055, pour cela nous avons dû installer toutes les bibliothèques nous permettant de pouvoir communiquer avec la centrale inertielle.

```
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
```

Ces bibliothèques sont facilement accessibles depuis le gestionnaire de librairies. Nous configurons ensuite l'adresse du bus I2C sur laquelle est branchée le BNO055 (0x28) :

```
Adafruit_BNO055 bno = Adafruit_BNO055(-1, 0x28);
```

Cette adresse est celle par défaut du capteur. Avec cette commande ci-dessous nous pouvons extraire les angles d'Euler du BNO055 :

```
imu::Vector<3> euler = bno.getVector(Adafruit_BNO055::VECTOR_EULER);
```

Nous récupérons un vecteur nommé "euler" contenant Yaw, Pitch et Roll. Nous décidons par la suite d'afficher ces angles dans le traceur série d'Arduino afin d'obtenir une idée du profil des données que nous aurons en sortie de l'interface que nous construirons par la suite.

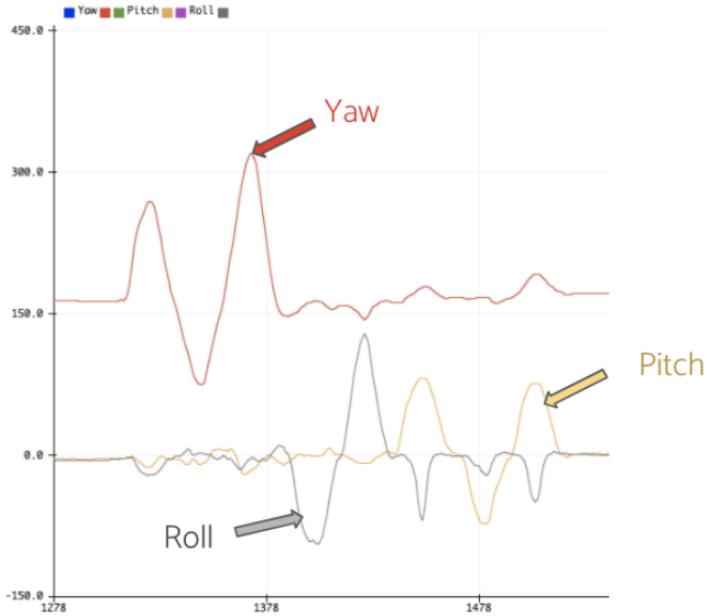


FIGURE 28 – Visualisation des angles d'Euler sur IDE Arduino

3.4.3 Choix du protocole sans fil

Après avoir réussi à extraire les angles d'Euler via le port série, nous devons désormais penser à la manière dont nous allons les envoyer via un protocole sans fil. Deux choix s'offraient à nous pour le protocole :

- ZigBee ;
- BLE (Bluetooth Low Energy).

Le choix a vite été pris entre ces deux protocoles car notre Arduino Nano 33 BLE, comme son nom l'indique, comporte une antenne BLE et donc notre choix s'est tout de suite rabattu sur le BLE afin de gagner en place et en coût sur nos capteurs. Néanmoins si nous avions eu une autre carte que l'Arduino Nano 33 BLE, notre choix aurait resté inchangé car le BLE nous propose une portée bien plus élevée que ZigBee, à savoir 100m pour le BLE et seulement 10m pour ZigBee. Une portée de 10m seulement aurait pu être un handicap lors de l'analyse de la marche d'un individu.



Nous allons commencer par expliquer le fonctionnement du BLE.

Le BLE est optimisé pour une utilisation à faible consommation d'énergie à de faibles débits de données et a été conçu pour fonctionner à partir de simples piles bouton au lithium.

Chaque élément connecté en BLE fait office soit de transmetteur soit de récepteur.
 Si notre carte est un transmetteur (appelée périphérique en langage BLE), elle envoie des données que l'ensemble des cartes connectées en BLE peuvent lire.
 Si notre carte est un récepteur (appelée centrale en langage BLE), elle lit à partir des données de l'un des périphériques qui contient des informations dont elle se soucie.

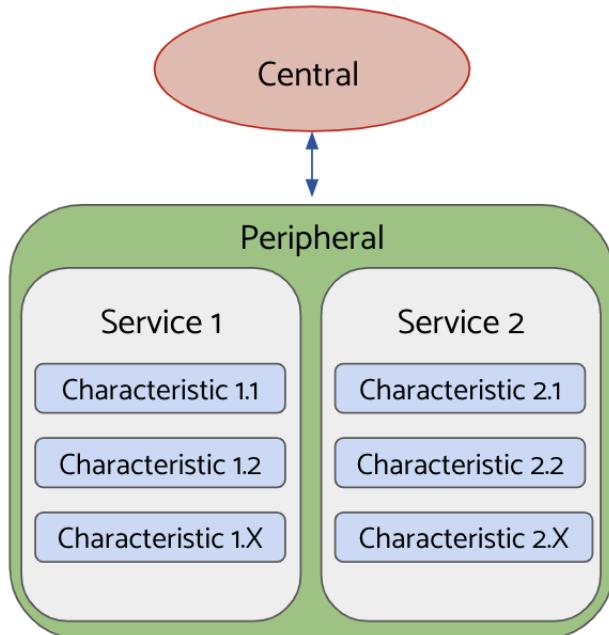


FIGURE 29 – Schéma de fonctionnement du BLE

Les informations présentées par un périphérique sont structurées en services, chacun d'eux étant subdivisé en caractéristiques. Une caractéristique peut être une température, du temps, une valeur de pression... En tant que périphérique, il suffit de mettre à jour chaque caractéristique du service voulu lorsqu'elle doit être mise à jour. En tant que centrale, il suffit de se connecter au périphérique voulu, puis de lire la caractéristique souhaitée.

Chaque service et caractéristique est distingué par un UUID (Universally Unique Identifier) codé sur 128 bits. À l'opposé des services qui lorsqu'ils sont identiques doivent avoir un UUID propre, deux caractéristiques de même nature envoyées dans deux services ou par deux périphériques différents peuvent avoir le même UUID.

Voici donc ci-dessous la structure de notre connexion BLE :

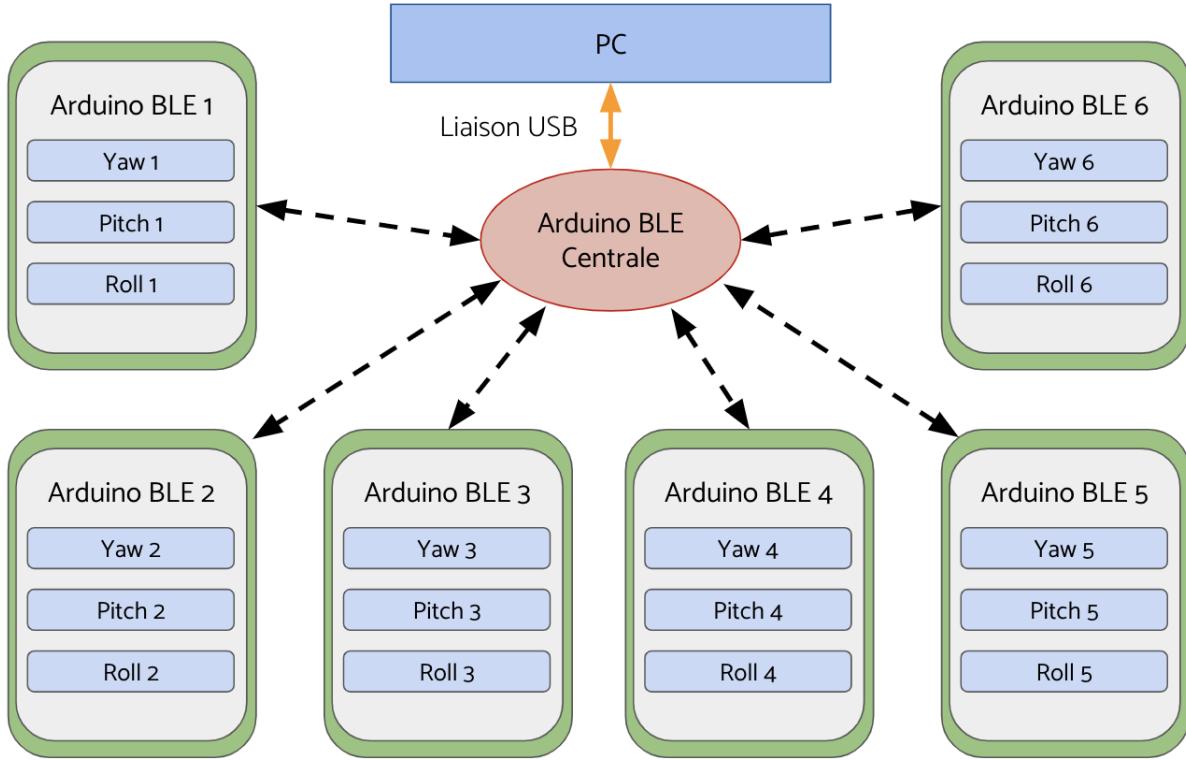


FIGURE 30 – Structure de connexion BLE

L'ensemble de nos 6 Arduino Nano 33 BLE périphériques seront reliées via BLE à une Arduino centrale, laquelle sera reliée via liaison USB au PC.

Chaque périphérique contiendra un seul service dans lequel seront regroupées trois caractéristiques que sont les angles d'Euler. Comme dit précédemment chaque service sera différencié par un UUID propre, tandis que chaque caractéristique "Yaw" portera le même UUID. Il en sera de même pour les caractéristiques "Pitch" et "Roll".

Afin de structurer les données reçues par la centrale et les envoyer correctement via le port série, nous avons dû organiser les données avec une trame d'envoi. Cette trame comporte plusieurs sous-parties.

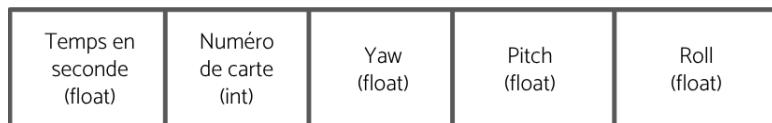


FIGURE 31 – Structure de la trame envoyée via le port série

La première partie sert de repère temporel afin de savoir à quel moment l'acquisition a été réalisée. Puis nous avons un identifiant qui nous indique le numéro de la carte qui nous envoie les données (allant de 1 à 7). Finalement les trois dernières parties de la trame sont réservées aux angles d'Euler que la carte en question nous renvoie.

Cette trame sera envoyée puis décomposée par le traitement qui sera réalisé pour l'interface python.

3.4.4 Contraintes de l'Arduino et de la multi connexion BLE

Étant sous la contrainte de devoir connecter un grand nombre de cartes à une seule centrale, nous nous sommes posés la question afin de savoir comment procéder pour pouvoir connecter toutes les cartes à une seule centrale.

Nous avons dans un premier temps pensé à établir une multi connexion avec les 6 périphériques de manière simultanée, c'est-à-dire que l'ensemble de tous les périphériques soient connectés en même temps. Cette solution semblait envisageable car l'antenne BLE présente sur l'Arduino (NINA B306) peut gérer au moins 8 connexions simultanées pour une seule centrale, néanmoins la bibliothèque BLE Arduino n'implémentait pas les fonctions nécessaires au multi connexion. Nous avions dans un premier temps essayé le multi connexion à l'aide d'une bibliothèque BLE qui avait été modifiée par un utilisateur afin de proposer des connexions simultanées, malheureusement les tests réalisés avec cette bibliothèque ce sont montrés non-concluants.

C'est de ce constat que nous avons décidé de changer de stratégie et de passer sur une méthode de connexion/déconnexion séquentielle. Cette méthode inclue ses avantages et ses inconvénients face au multi connexion.

	Multi connexion	Connexion séquentielle
Avantages	Fréquence d'acquisition élevée	Gestion/Connexion plus simple aux périphériques
Inconvénients	Difficile à mettre en place sur Arduino	Fréquence d'acquisition trop faible Connexions aléatoires

FIGURE 32 – Comparaison multi connexion et connexion séquentielle BLE

Le principal désavantage de la connexion séquentielle est la fréquence d'acquisition à laquelle nous récupérons les données de chaque périphérique.

3.5 Réalisation de l'interface sous python

Pour traiter et analyser nos données, nous devons réaliser une interface nous permettant d'observer nos angles d'Euler sur des courbes et de pouvoir enregistrer ces données pour qu'elles puissent être traitées par les utilisateurs.

Nous hésitions entre deux supports pour la réalisation de notre interface : Matlab ou Python. Notre choix s'est porté sur Python (version 3.9.2) car c'est un langage de programmation très utilisé et son utilisation est gratuite contrairement à Matlab, ce qui s'inscrit dans la démarche de réduire les coûts de notre solution.

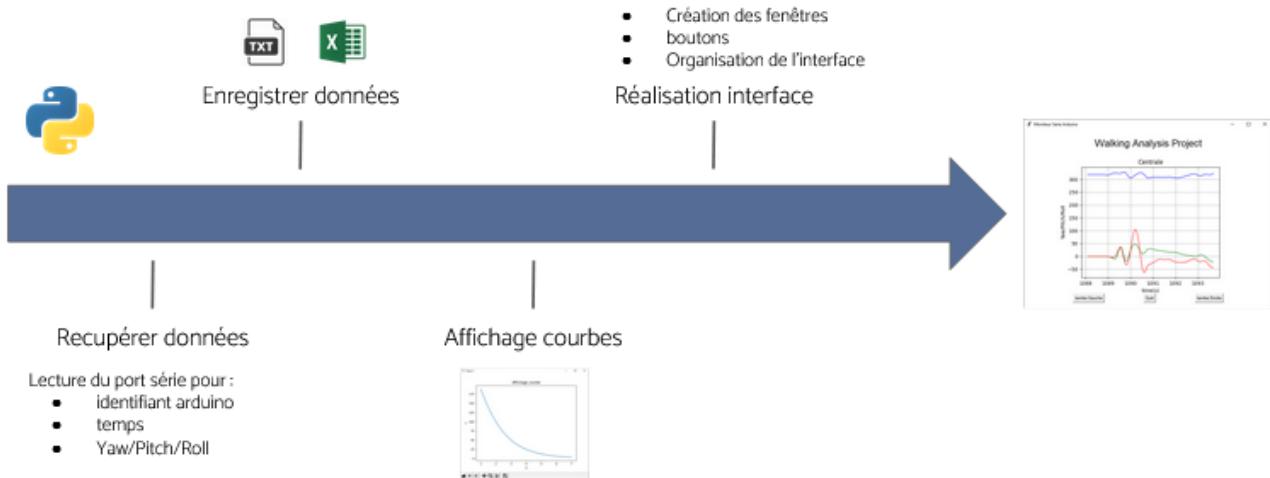


FIGURE 33 – Frise chronologique de développement réalisé

Nous allons voir les différentes étapes de développement comme présenté sur la Figure 33 ci-dessus qui sont :

- Récupérer les données ;
- Enregistrer les données pour être traité ;
- Afficher les courbes pour l'interprétation ;
- Réaliser l'interface finale.

3.5.1 Récupération des données

La première étape a été de communiquer avec l'Arduino à partir de notre code Python. Pour cela, il faut dans un premier temps pouvoir lire nos ports séries grâce à la librairie **PySerial**.

Une fois que nous avons réussi à nous connecter au port série, il faut récupérer les trames de données envoyées depuis l'Arduino (temps/identifiant arduino/Yaw/Pitch/Roll), et "spliter" notre trame pour que chaque élément soit mis dans un tableau. A chaque acquisition, nous ajoutons dans notre tableau nos nouvelles valeurs.

3.5.2 Enregistrement des données

Une fois que nous avons réussi à récupérer nos données, nous voulons pouvoir les enregistrer. Enregistrer les données permet que celle-ci soit ré-exploitable une fois le programme fini.

Nous enregistrons nos données dans un *fichier.txt* pour que celui-ci puisse être ouvert sur Excel par exemple.

	id	arduino	temp(s)	Yaw	Pitch	Roll
1	5.0	22.78	216.5	1.94	-7.5	
2	7.0	22.78	2.11	41.31	-91.12	
3	7.0	25.21	5.37	41.69	-93.62	
4	7.0	25.5	4.75	42.81	-92.44	
5	4.0	25.8	193.25	18.94	-36.75	
6	7.0	25.8	3.5	6.31	27.31	
7	6.0	27.45	255.69	-11.63	5.56	
8	7.0	27.45	14.88	4.87	47.38	
9	4.0	28.85	168.69	16.37	-83.44	
10	6.0	28.85	242.31	-39.0	-13.25	
11	7.0	28.85	14.38	34.13	-40.63	
12	3.0	31.25	116.06	20.19	53.19	
13	7.0	31.25	11.5	-10.75	42.88	
14	7.0	32.75	3.44	-19.19	45.75	
15	7.0	33.04	1.37	-13.31	21.12	
16	2.0	33.3	338.19	-37.44	-100.19	
17	7.0	33.3	10.81	27.25	-21.37	
18	1.0	34.62	39.75	47.88	-17.25	

FIGURE 34 – Exemple de *fichier.txt* d'acquisition

Voici un exemple ci-dessus sur la Figure 34 d'un fichier que nous avons enregistré (au nom de "acquisition.txt"). Nous pouvons observer à chaque ligne une nouvelle acquisition et que chaque éléments dans nos trames sont séparés par un espace.

Le choix du nom du fichier se fait à chaque lancement de notre code. Tant que notre code ne reçoit pas un nom de fichier en entrée, l'acquisition ne peut pas se lancer.

Une fois notre *fichier.txt* enregistré, nous pouvons l'ouvrir sur Excel en paramétrant que chaque valeur dans le fichier est séparé par un espace. Nous obtenons le résultat ci-dessous sur la Figure 35.

	A	B	C	D	E
1	5.0	22.78	216.5	1.94	-7.5
2	7.0	22.78	2.81	41.31	-91.12
3	7.0	25.21	5.37	41.69	-93.62
4	7.0	25.5	4.75	42.81	-92.44
5	4.0	25.8	193.25	18.94	-36.75
6	7.0	25.8	3.5	6.31	27.31
7	6.0	27.45	255.69	-11.63	5.56
8	7.0	27.45	14.88	4.87	47.38
9	4.0	28.85	168.69	16.37	-83.44
10	6.0	28.85	242.31	-39.0	-13.25
11	7.0	28.85	14.38	34.13	-40.63
12	3.0	31.25	116.06	20.19	53.19

FIGURE 35 – Exemple de *fichier.txt* ouvert avec Excel

3.5.3 Affichage courbes

La troisième étape de développement a été de tracer sur des courbes nos valeurs de Yaw, de Pitch et de Roll par rapport au temps. Pour l'affichage de nos courbes, nous avons décidé d'utiliser la librairie **matplotlib**.

Nous avons opté pour 2 stratégies d'acquisitions :

- Un affichage après traitement des données ;
- Un affichage en temps réel.

La première solution étant la plus facile, nous allons commencer par expliquer celle-ci. Comme décrit dans la partie "*Récupérer les données*", nous avons nos différentes valeurs remplies dans des tableaux.

Nous affichons le temps à laquelle s'est effectuée l'acquisition en abscisse et nous affichons le Yaw, le Pitch et le Roll en ordonnée (le programme correspondant sur le github est celui nommé **data_save.py**).

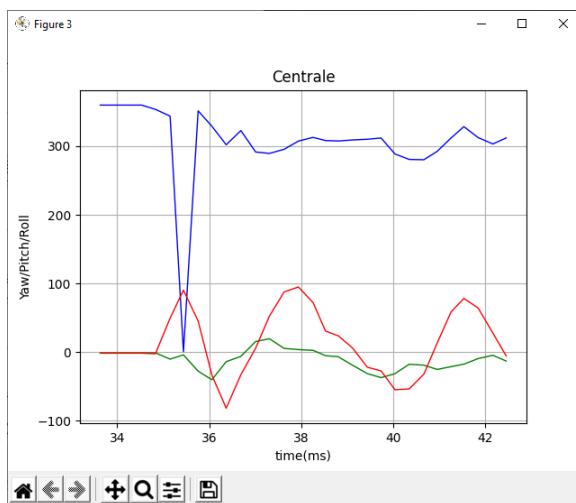


FIGURE 36 – Affichage courbe avec traitement

Pour que l'interface s'affiche, il faut d'abord réaliser notre acquisition complètement et c'est une fois celle-ci réalisée que nous pouvons afficher nos courbes.

Dans l'exemple ci-dessus nous affichons uniquement la centrale, cependant nous pouvons afficher tous les segments en fonction de l'identifiant arduino envoyé.

La deuxième solution est d'afficher nos courbes en temps réel (le programme correspondant sur le github est celui nommé **serie.py**).

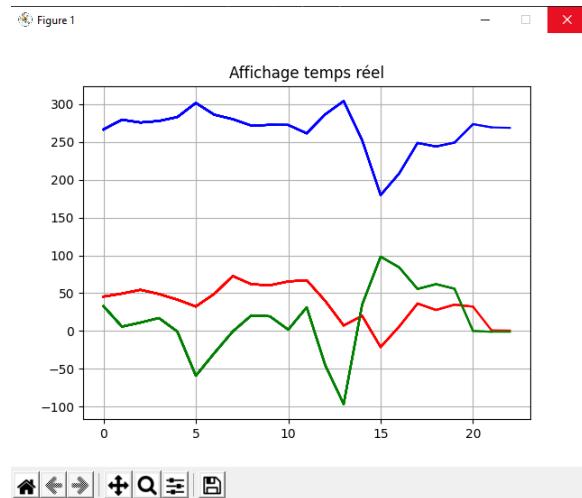


FIGURE 37 – Affichage courbe en temps réel

Lorsque nous affichons nos courbes en temps réel au vu du grand nombre de données enregistrées, matplotlib a du mal à afficher les courbes car il y a un très grand nombre de valeurs à traiter. Sur nos machines, l'affichage fonctionnait uniquement pour une acquisitions de quelques secondes.

Cette solution n'est pas idéale pour le moment, mais elle reste une option possible de développement.

3.5.4 Réalisation interface

La dernière étape de développement a été de réaliser l'interface et d'intégrer toutes les phases de développement expliquées précédemment dans celle-ci.

Pour la réalisation de cette IHM (Interface Homme-Machine), nous utilisons la librairie **Tkinter**.

Les premières étapes de développement ont été de créer l'interface et de la designer. L'idée est que sur notre interface nous affichons la "centrale" (Arduino situé sur le bassin) dans une fenêtre, puis l'aide de **boutons** nous pouvons souhaiter d'afficher les segments de la jambe gauche, les segments de la jambe droite, ou bien de fermer l'application.

Nous pouvons observer ci-dessous un exemple de notre interface réalisée.

Pour afficher nos courbes à l'intérieur de notre interface, il a fallu intégrer matplotlib dans Tkinter.

Pour l'affichage des courbes, nous n'avons pas choisi de faire du temps réel car cette solution présentait déjà quelques problèmes et l'intégration de celle-ci dans notre interface générait beaucoup de problèmes et un temps de développement allongé. C'est pourquoi nous avons décidé d'utiliser la solution où nous enregistrons dans un premier temps nos données, puis nous les affichons une fois l'acquisition terminée.

Une autre amélioration effectuée dans cette partie a été de lisser la courbe de la centrale.

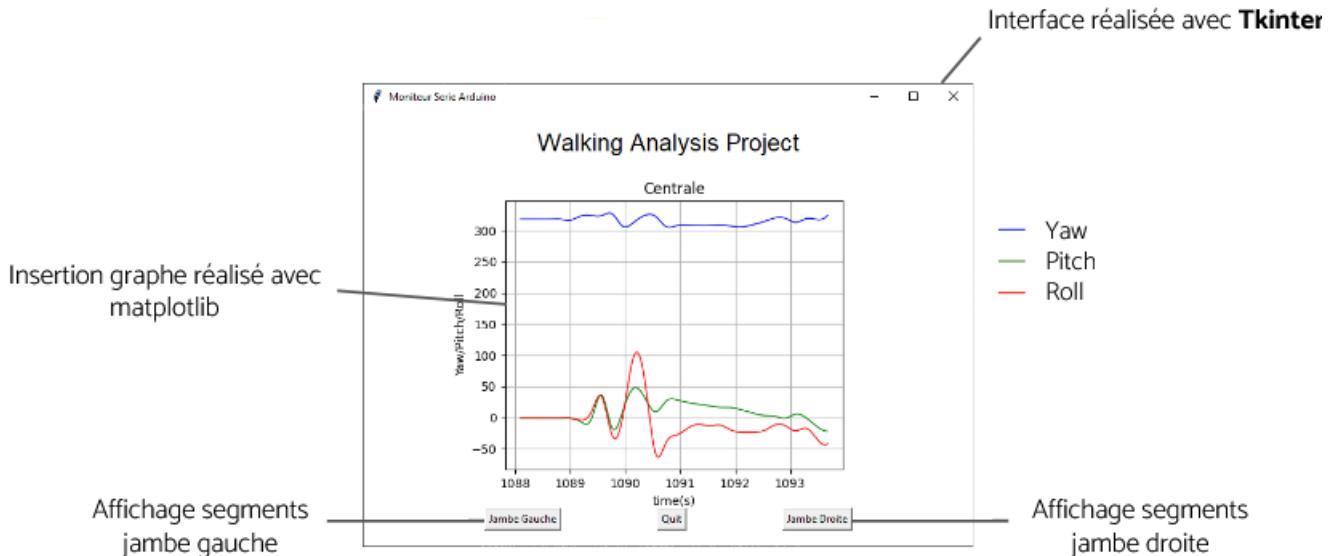


FIGURE 38 – Affichage interface terminée

Lorsque nous cliquons sur l'un des deux boutons situé à droite ou à gauche de l'interface, nous pouvons afficher la jambe que nous souhaitons comme ci-dessous :

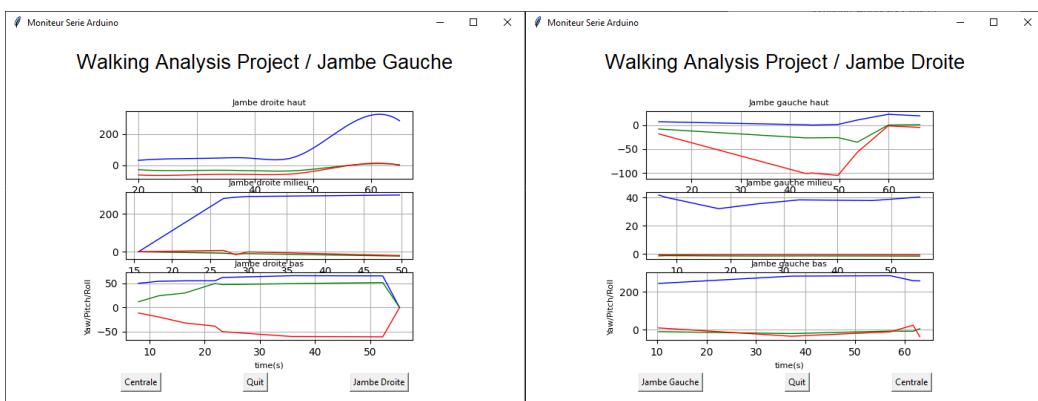


FIGURE 39 – Affichage interface segments gauche/droite

Notre interface fonctionne et répond à ce que nous souhaitions. Nos données sont enregistrées dans un fichier et chaque segment est affiché sur notre interface pour pouvoir être interprété.

Cependant, nous pouvons trouver quelques axes d'améliorations. Comme énoncé précédemment, nous pourrions réussir à faire un affichage temps réel, mais nous pourrions aussi ajouter d'autres boutons comme par exemple mettre l'acquisition en pause, ou bien choisir d'enregistrer le fichier à partir de notre interface.

4 Conclusion

Dans cette partie, nous conclurons dans un premier temps sur notre solution technique, c'est-à-dire sur la réponse au cahier des charges, les problèmes rencontrés et les optimisations que nous avons identifiées. Nous finirons sur ce que nous a apporté ce projet d'un point de vue connaissance et personnel.

4.1 Conclusion technique

4.1.1 Réponse au cahier des charges

Finalement lors de ce projet, nous avons réussi à répondre à la majorité des exigences du cahier des charges. Nous avons réussi à créer un prototype sans fil à l'aide du protocole BLE permettant de faire communiquer chaque capteurs périphériques à un capteur centrale. Pour ce qui est de la fréquence d'acquisition de 100Hz, elle n'a pas pu être respectée en raison des problèmes que nous avons rencontré avec notre code de la centrale Arduino. La précision angulaire de 5° à bien été respectée en choisissant une centrale inertielle (BNO055) qui offre des données en sortie stables et sans dérive dans le temps.

Nous avons par la suite pu afficher et traiter l'ensemble des données via une interface python. Et enfin nous avons rendu ce projet utilisable avec la conception de PCBs sur Eagle et de boîtiers réalisés via une conception de modèles 3D.

Le coût total du projet sans PCB nous est revenu à 495 €.

4.1.2 Problèmes rencontrés

Lors de la conception de notre dispositif nous avons rencontré plusieurs problèmes, dans la partie software comme dans la partie hardware. Dans cette partie nous allons revenir sur les problèmes rencontrés.

Partie software

Pour ce qui est de la partie software, les problèmes sont les suivants :

- Connexion avec la carte STM ;
- Multi-connexion des cartes Arduino ;
- Environnement de développement Python.

Les premiers problèmes rencontrés ont été avec la carte STM que nous avions tout d'abord choisie pour faire la collecte des données des cartes Arduino et qui était supposée transmettre les données collectées au PC en Bluetooth. Lors de nos différents tests nous avons réussi à interagir avec la carte STM par l'intermédiaire d'un smartphone. Mais très vite nous nous sommes heurté au problème de la connexion avec le PC.

Comme expliqué précédemment, nous avons rencontrés des problèmes avec la multi-connexion des périphériques Arduino à la centrale. Nous avons réussi à outrepasser ce problème en établissant une connexion séquentielle au prix d'une fréquence d'acquisition qui ne respectait plus le cahier des charges. Avant cette solution de connexion séquentielle qui nous a été proposée par Mme Abir REZGUI, nous étions résolus à réussir la multi-connexion. Par conséquent nous avons essayé de remplacer la carte Arduino Nano 33 BLE centrale par une carte de chez STM, la STM32WB55 qui proposait dans son software un développement natif d'une multi-connexion d'une centrale vers plusieurs périphériques.

Finalement nous sommes restés sur notre choix de base qui est l'Arduino, ce choix nous a arrangé aussi afin de pouvoir uniformiser notre interface de travail sur un seul environnement.

L'environnement de développement de Python a aussi été un frein en début du projet. Nous voulions de base travailler sur une machine virtuelle sous Linux, cependant la communication avec les ports série posait problème. Nous avons voulu installer Linux en dual boot, or l'installation ne fonctionnait pas car un ancien dual boot avait été mal désinstallé.

Nous avons donc fait le choix de continuer à développer sous Windows avec Visual Studio Code comme IDE. Ce problème nous a fait perdre une journée complète de développement.

Partie hardware

Pour ce qui est de la partie hardware, les problèmes sont les suivants :

- Déconnexion intempestive des cartes Arduino ;
- Erreur de conception PCB.

En effet, lors de nos tests nous nous sommes souvent heurté à des problèmes avec les cartes Arduino. Celles-ci se déconnectaient sans raisons apparentes.

Les problèmes rencontrés vis-à-vis du PCB venait d'un problème de conception. Une fois les différents problèmes analysés (faux contacts, mauvaises liaisons...), nous avons pu les corriger en soudant des nouvelles connexions.

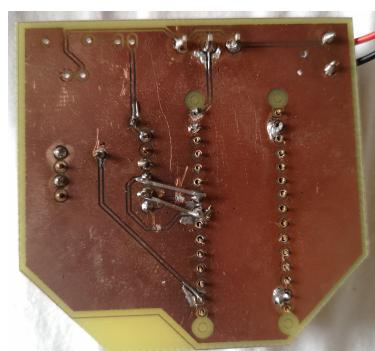


FIGURE 40 – Soudures

On remarque sur la Figure 40 que des fils ont été soudés entre les différents pins de la carte afin de résoudre le problème de communication que nous rencontrions entre la carte Arduino et le BNO055.

4.1.3 Optimisations possibles

A la fin de ce projet, nous avons identifiés 4 optimisations possibles pour mieux répondre au cahier des charges et pour rendre l'expérience utilisateur plus agréable.

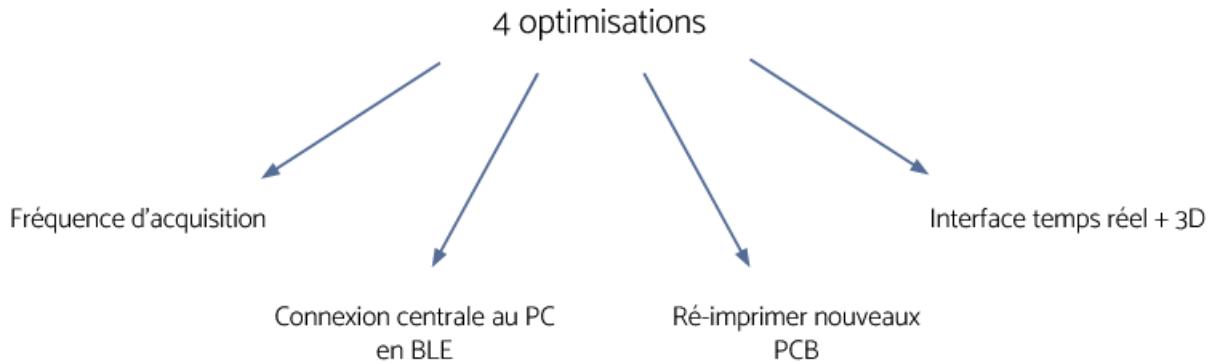


FIGURE 41 – Optimisations identifiées

Optimisation n°1

La première optimisation est celle de la fréquence d'acquisition qui est notre seul problème lié au cahier des charges.

La première solution identifiée pour palier à ce problème est de faire du "multiconnexion" en BLE car actuellement la connexion de nos cartes en séquentiel prend énormément de temps. La deuxième solution identifiée est d'optimiser notre code Arduino et notre code Python afin que chaque instruction prennent le moins de temps et nous permettent de réduire notre temps d'acquisition.

Optimisation n°2

La seconde optimisation est de connecter notre centrale et notre PC en BLE. Actuellement, les deux sont connectés en filaire, ce qui implique que lorsque nous faisons une acquisition, le PC doit se déplacer en même temps que notre individu testé.

Optimisation n°3

Notre troisième optimisation est de ré-imprimer les PCB. Ceux-ci possédant des défaillances expliqués précédemment, nous devons les ré-imprimer pour que notre solution soit validée.

Optimisation n°4

Notre dernière optimisation identifiée est d'intégrer une interface temps réel qui nous affiche les courbes "en direct" lors de l'acquisition.

Une autre optimisation du code Python serait d'ajouter une interface en 3 dimensions.

4.2 Conclusion personnelle

Tout d'abord, lors de ce projet nous avons pu développer et approfondir nos connaissances techniques. Ces connaissances ont été :

- La découverte du protocole BLE ;
- La technologie d'un IMU ;
- Le développement sur Arduino ;
- Le langage Python ainsi que les librairies utilisées ;
- La création de PCB avec Eagle ;
- La réalisation 3D avec SketchUp ;
- Le suivi de version avec Github.

De plus, pour nous organiser, nous avons fait appel à des techniques de gestion de projet comme le diagramme de Gantt pour répartir les tâches et avoir une vue d'ensemble des livrables à réaliser.

La crise du Covid-19 durant tout le projet nous a aussi appris à trouver de nouveaux moyens de travailler ensemble à distance, comme la mise en place de réunion assez fréquente pour faire des points sur les avancées et les développements réalisés.

La réalisation de ce projet nous aura aussi énormément aidé à développer notre communication et le travail en équipe. Chaque voix a pu être écoutée et débattue, les rapports entre nous ce sont relativement bien passé.

Un des points les plus importants de cette expérience à été notre prise de conscience de la complexité d'un projet. Nous avons remarqué qu'il y a toujours des imprévus, cela demande de l'anticipation dès le début du projet, notamment dans le diagramme de Gantt. Nous en avons tiré des leçons pour mieux comprendre la vie d'un projet et ainsi être plus performants pour nos futurs projets professionnels.

Pour conclure, nous avons apprécié réaliser ce projet entre camarades de classes pour clôturer notre cursus scolaire à ESIEE Paris. Ce projet s'inscrit dans une étape importante de notre scolarité et de notre vie : la transition entre la vie étudiante et la vie professionnelle.

5 Bibliographie

Arduino site, <https://www.arduino.cc>

Arduino blog, <https://blog.arduino.cc>

STMicroelectronics, <https://www.st.com/content/stcom/en.html>

Stack Overflow, <https://stackoverflow.com>

Matplotlib, <https://matplotlib.org>

HAYOT Chris, Analyse biomécanique 3D de la marche humaine : Comparaison des modèles mécaniques, 2006, <http://nuxeo.edel.univ-poitiers.fr/nuxeo/site/esupversions/79a24c44-5587-4172-abb4-2eb7651d6ce5>

ARMAND Stéphane, Analyse Quantifiée de la Marche : extraction de connaissances à partir de données pour l'aide à l'interprétation clinique de la marche digitigrade, 2005, <https://tel.archives-ouvertes.fr/tel-00010618/document>

Quelle technologie sans fil utiliser pour fabriquer des objets connectés DIY, <https://projetsdiy.fr/quelle-technologie-sans-fil-objets-connectes-diy/>

Getting Started with Bluetooth LE on the Arduino Nano 33 Sense, <https://ladvien.com/arduino-nano-33-bluetooth-low-energy-setup/>

STM32WB BLE stack programming guidelines, https://www.st.com/resource/en/programming_manual/dm_stm32wb-ble-stack-programming-guidelines_stmicroelectronics.pdf

Adafruit BNO055 Absolute Orientation Sensor, <https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor?view=all>

Communication of One Central Device and More than One Peripheral Device, <https://github.com/arduino-libraries/ArduinoBLE/issues/50>