PROJET ARDUINO – PEIP2

Année scolaire 2018-2019

"Casque de vélo intelligent"

Etudiants : STRABACH Margot, MASSON Paul

Encadrant : Mr. MASSON

SOMMAIRE

1 INTRODUCTION

Le but de ce projet est de développer un casque de vélo, dit intelligent, permettant d'augmenter la visibilité du cycliste sur la voie publique.

En effet, nous avons choisi comme point de départ à ce projet la question suivante : pourquoi le vélo est-il si peu utilisé comme moyen de transport quotidien ? Malgré les incitations et la tendance à favoriser et développer les moyens de transports éco-responsables, le vélo reste très peu utilisé face aux automobiles, motos et scooters. La raison principale réside dans le fait que ce mode de déplacement présente trop d'inconvénients par rapport aux bénéfices qu'il apporte. Jugé par la plupart comme trop lent, fatigant ou encore peu sécurisé, il est plus vu comme un divertissement plutôt qu'un réel moyen de transport.

Pour remédier à cela, le vélo a su se réinventer afin de répondre aux besoins actuels. Il possède aujourd'hui une assistance électrique permettant d'augmenter considérablement sa vitesse tout en réduisant l'effort fourni par l'utilisateur. Cependant, il reste bien moins sécurisé que les autres moyens de transports. Au-delà de l'insuffisance de protections physiques par rapport aux autres véhicules, le manque de visibilité est la raison principale à ce problème. Nous avons donc axé notre projet autour de cette problématique :

Comment améliorer la visibilité des cyclistes ?

Pour y répondre nous avons eu l'idée d'adapter les équipements de signalisation d'un véhicule motorisé à un casque de vélo, en lui ajoutant des clignotants, un feu de stop, des lumières pour être vu et un phare pour éclairer la route. L'ensemble de ces éléments permet de faciliter le partage de la route en rendant les intentions et mouvements du cycliste plus prévisibles pour les autres usagers.

La première étape de ce projet consiste à rédiger un cahier des charges afin de définir les besoins et les contraintes de ce nouveau casque. Une fois ces fonctions déterminées, le projet est décomposé en deux grandes parties nommées 'les contrôles du casque' et 'l'affichage sur le casque', illustrées par la figure ci-dessous.

Figure 1: *Le casque de vélo intelligent en situation*

La première partie regroupe toutes les commandes et informations auxquelles le cycliste doit avoir accès, ainsi que la détection du freinage et l'échange des données avec le casque.

La deuxième comporte le traitement des informations reçues (clignotants, phare, freinage), leurs affichages sur le casque, la détection de la luminosité afin de rendre le cycliste visible la nuit et la transmission de l'état lumineux du casque à l'attention du cycliste.

2 ANALYSE FONCTIONNELLE DU BESOIN

La figure ci-dessous représente les principales fonctions de notre projet.

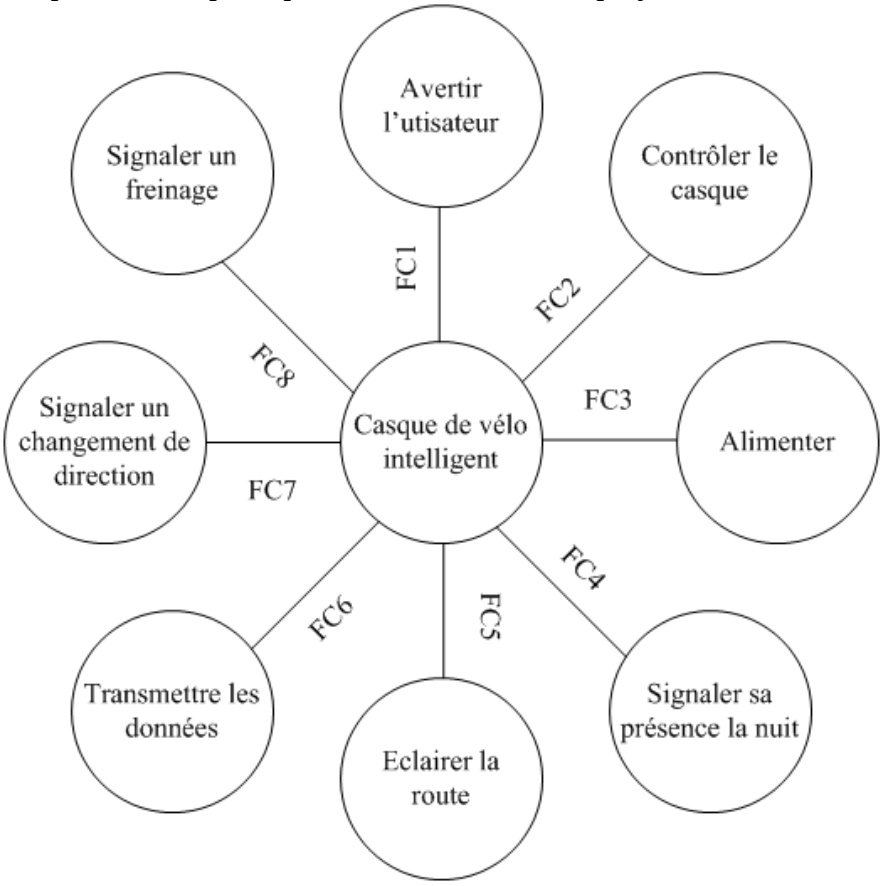


Figure 2: *Diagramme pieuvre des principales fonctions*

Fonctions	Noms	Contraintes
FC1	Avertir l'utilisateur	L'utilisateur doit connaître l'état de son casque pour savoir si un de ses clignotants, son éclairage ou son feu de stop sont allumés. Ces indications doivent être visibles sur le guidon du vélo.
FC2	Contrôler le casque	Les commandes doivent être accessibles en roulant, à proximité immédiate des mains du cycliste. Elles doivent donc se trouver sur le guidon. Il doit y avoir une commande pour les clignotants et une autre pour le phare.
FC3	Alimenter	Le casque et ses contrôles doivent être autonomes en énergie. Les sources d'alimentation doivent être embarquées sur le casque et le vélo.
FC4	Signaler sa présence la nuit	Il faut mesurer le niveau de luminosité afin d'allumer automatiquement l'éclairage du casque. Cette détection doit se faire sur le casque afin d'accompagner le cycliste s'il quitte son vélo.
FC5	Eclairer	La route devant l'utilisateur doit être éclairée d'une portée suffisante pour circuler la nuit.
FC6	Transmettre les données	La transmission doit être sans fil.
FC7	Signaler un changement de direction	La signalisation doit se trouver à l'avant et à l'arrière du casque.
FC8	Signaler un freinage	La signalisation doit se trouver à l'arrière du casque. Il faut détecter si le vélo ralentit.

3 PRINCIPE DE FONCTIONNEMENT

La figure ci-dessous représente le fonctionnement du casque et de ses contrôles, ainsi que les principales ressources utilisées pour chaque étape.

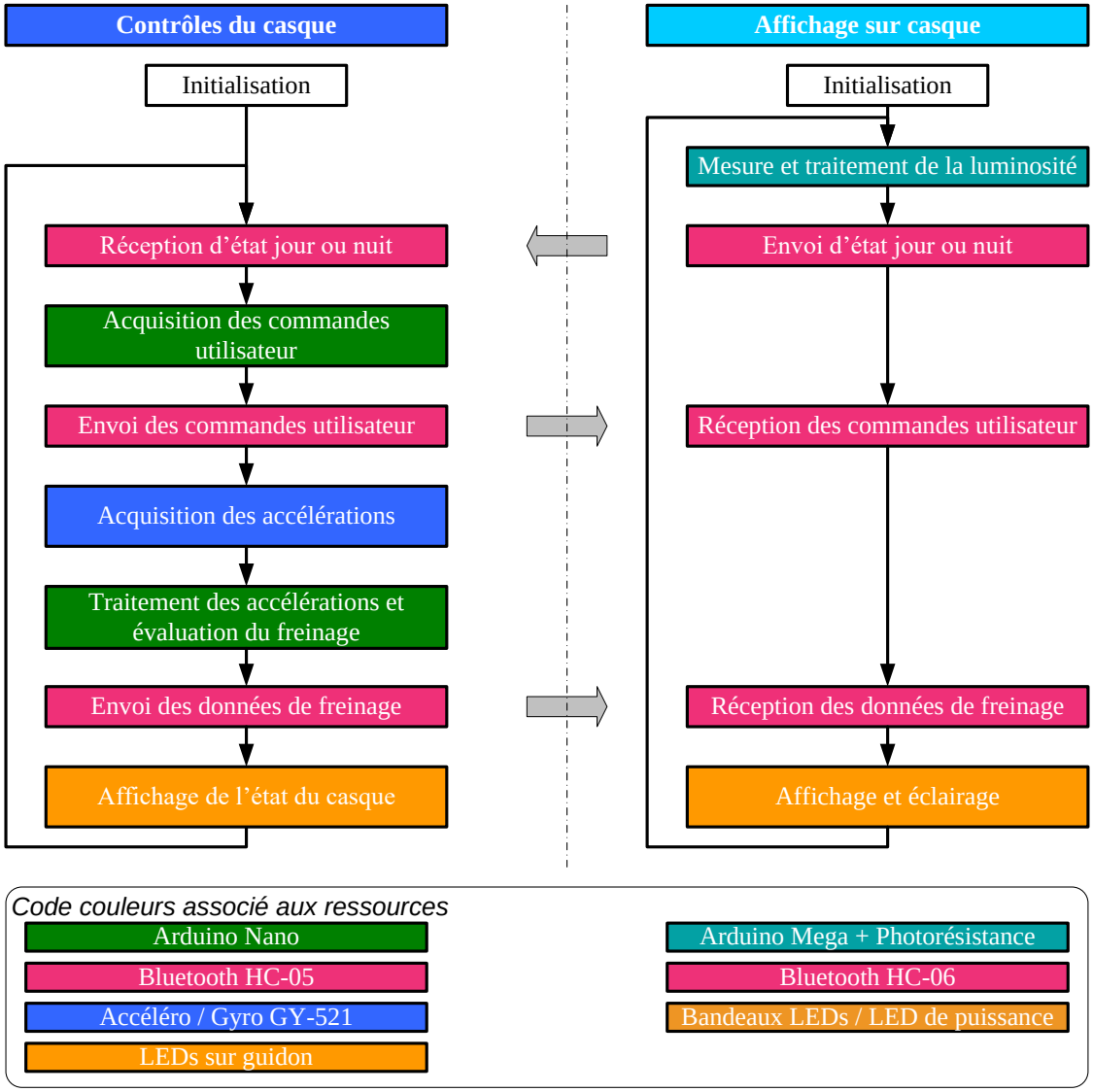
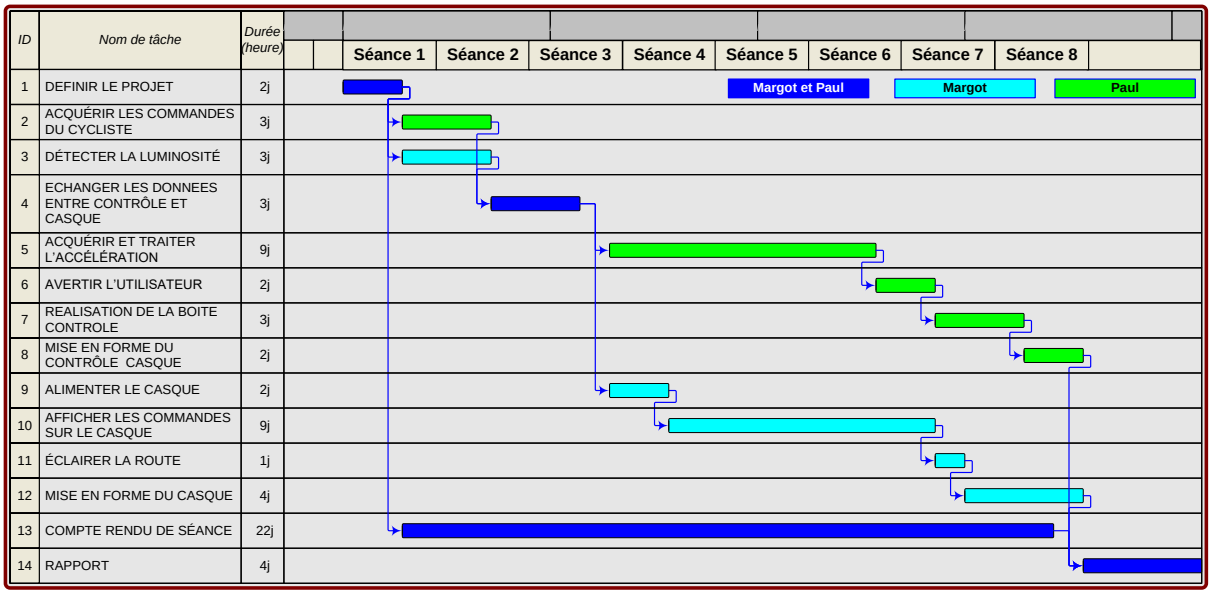


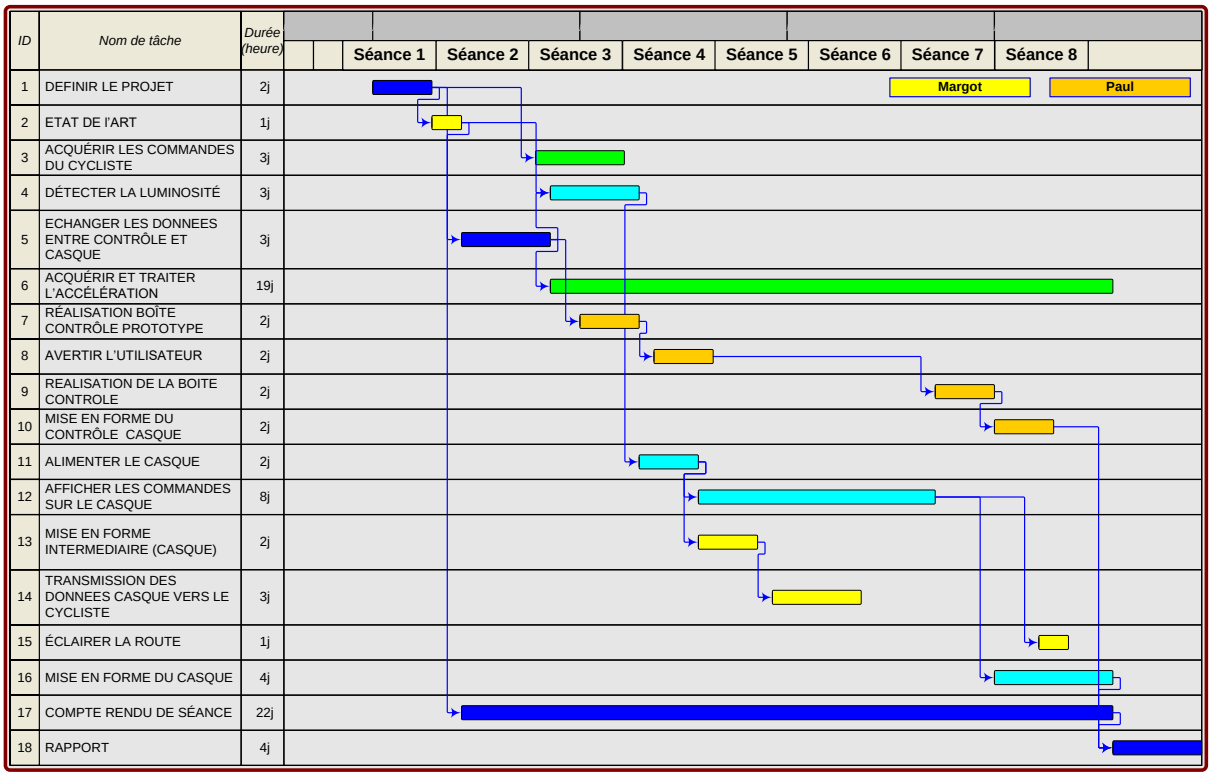
Figure 3: *Principe de fonctionnement du casque et de ses contrôles*

4 PLANIFICATION DES TACHES



ID	Nom de tâche	Durée (heure)
1	DEFINIR LE PROJET	2j
2	ACQUÉRIR LES COMMANDES DU CYCLISTE	3j
3	DÉTECTER LA LUMINOSITÉ	3j
4	ECHANGER LES DONNEES ENTRE CONTRÔLE ET CASQUE	3j
5	ACQUÉRIR ET TRAITER L'ACCÉLÉRATION	9j
6	AVERTIR L'UTILISATEUR	2j
7	REALISATION DE LA BOITE CONTROLE	3j
8	MISE EN FORME DU CONTRÔLE CASQUE	2j
9	ALIMENTER LE CASQUE	2j
10	AFFICHER LES COMMANDES SUR LE CASQUE	9j
11	ÉCLAIRER LA ROUTE	1j
12	MISE EN FORME DU CASQUE	4j
13	COMPTE RENDU DE SÉANCE	22j
14	RAPPORT	4j

Figure 4: *Diagramme de Gantt établi au début du projet*



ID	Nom de tâche	Durée (heure)
1	DEFINIR LE PROJET	2j
2	ETAT DE l'ART	1j
3	ACQUÉRIR LES COMMANDES DU CYCLISTE	3j
4	DÉTECTER LA LUMINOSITÉ	3j
5	ECHANGER LES DONNEES ENTRE CONTRÔLE ET CASQUE	3j
6	ACQUÉRIR ET TRAITER L'ACCÉLÉRATION	19j
7	RÉALISATION BOÎTE CONTRÔLE PROTOTYPE	2j
8	AVERTIR L'UTILISATEUR	2j
9	REALISATION DE LA BOITE CONTROLE	2j
10	MISE EN FORME DU CONTRÔLE CASQUE	2j
11	ALIMENTER LE CASQUE	2j
12	AFFICHER LES COMMANDES SUR LE CASQUE	8j
13	MISE EN FORME INTERMEDIAIRE (CASQUE)	2j
14	TRANSMISSION DES DONNEES CASQUE VERS LE CYCLISTE	3j
15	ÉCLAIRER LA ROUTE	1j
16	MISE EN FORME DU CASQUE	4j
17	COMPTE RENDU DE SÉANCE	22j
18	RAPPORT	4j

Figure 5: *Diagramme de Gantt des tâches réalisées*

Les diagrammes ci-dessus réalisés en début et fin de projet montrent que nous avons sous-estimé la quantité de travail et le temps nécessaire à la réalisation du traitement de l'accélération. En effet, celui-ci a nécessité deux fois plus de temps que prévu. Nous avons aussi rencontré des difficultés avec la transmission des données, ce qui nous a fait perdre une séance sur le planning initial. De plus, nous avons été contraints d'ajouter des tâches intermédiaires comme la création d'un premier boîtier ainsi qu'une première mise en forme du casque afin de pouvoir continuer efficacement notre travail. Finalement, les fonctions les plus faciles ont été réalisées en parallèles de celles plus complexes.

5 LES CONTROLES DU CASQUE

5.1 LES DIFFERENTES FONCTIONS

5.1.1 FONCTION FC2 : CONTROLER LE CASQUE

Afin de réaliser cette fonction « Contrôler le casque », nous avons utilisé un commutateur de scooter.

Figure 6: *Commutateur de scooter utilisé*

Celui-ci présente l'intérêt de fournir les commandes nécessaires au cahier des charges : un interrupteur à 3 positions pour les clignotants et un switch pour le phare. De plus, son ergonomie facilite grandement l'utilisation des commandes en roulant. Enfin, il est facilement installable sur le guidon de l'utilisateur, ce qui nous a permis d'économiser du temps puisque nous n'avions pas à concevoir cette interface.

Cette fonction exécute deux algorithmes séquentiels, l'un servant aux commandes des clignotants et l'autre à celle du phare. Ils ont été conçus afin que seuls les changements de commande soient transmis au casque. De ce fait, les communications entre le contrôle du casque et le casque lui-même sont limitées au strict nécessaire, ce qui permet de minimiser leurs consommations.

Les interrupteurs du commutateur sont lus par la carte Arduino au travers d'entrées digitales équipées de pull-down, de sorte que lorsqu'un interrupteur est activé la carte lit un '0' logique.

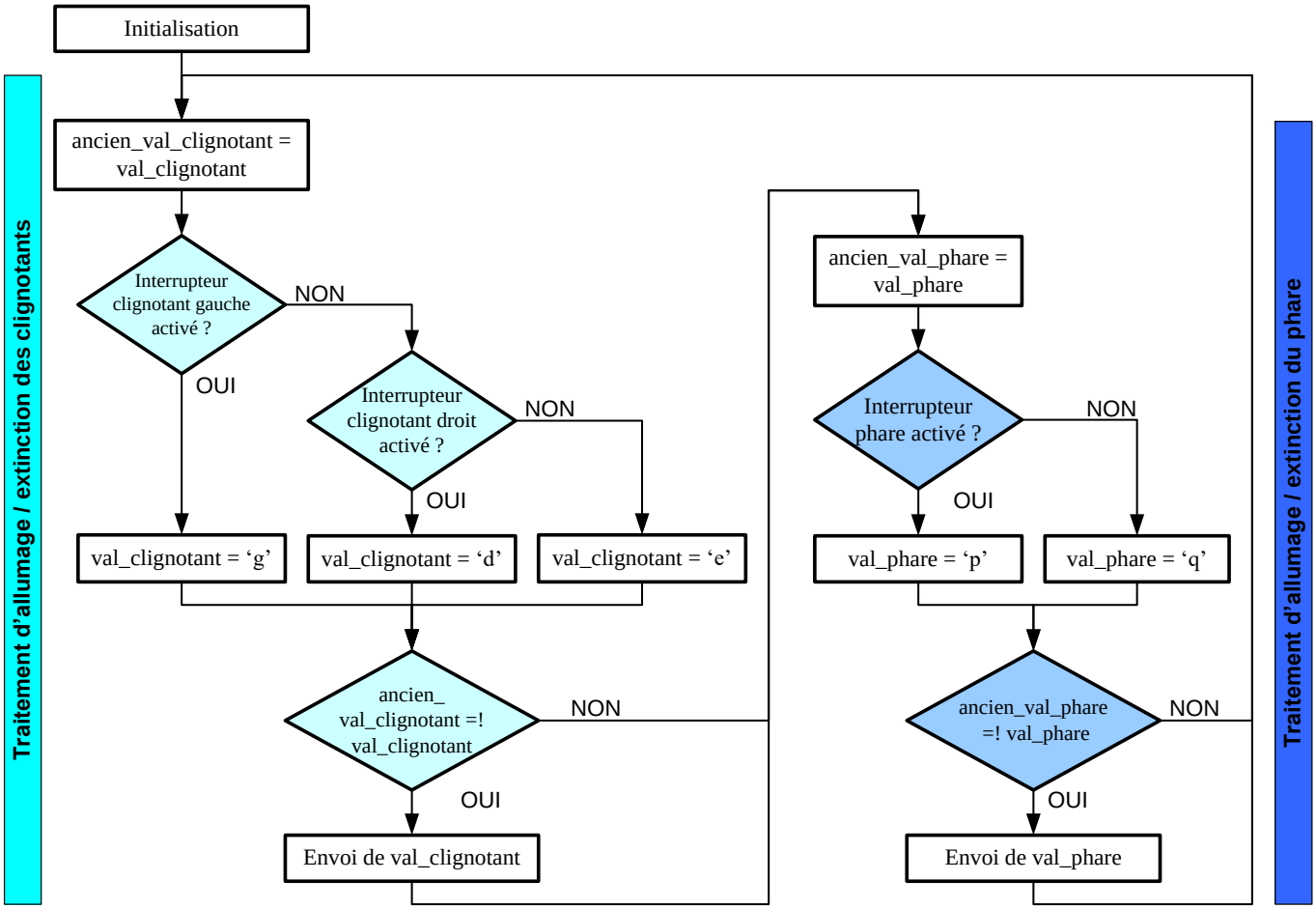


Figure 7: *Algorithmes de la fonction FC2*

5.1.2 FONCTION FC8 : SIGNALER UN FREINAGE

Afin d'accomplir la fonction « Signaler un freinage », il est indispensable de savoir si le cycliste freine. Pour obtenir cette information, nous avons choisi de mesurer l'accélération du vélo. Cette solution présente l'avantage de détecter sa décélération, quelque-soit l'action du cycliste : utilisation des freins ou simple ralentissement. Dans les deux cas, les autres usagers sont avertis du danger et le risque d'accident est réduit.

Pour mesurer cette accélération, nous avons retenu l'accéléromètre GY-521, car celui-ci dispose en plus de trois accéléromètres, d'un gyroscope 3 axes. Ce dernier permet de corriger l'erreur de mesure d'accélération due à l'inclinaison du vélo, suivant qu'il se situe en descente ou en montée.
Ce composant communique par liaison série I2C avec la carte Arduino.

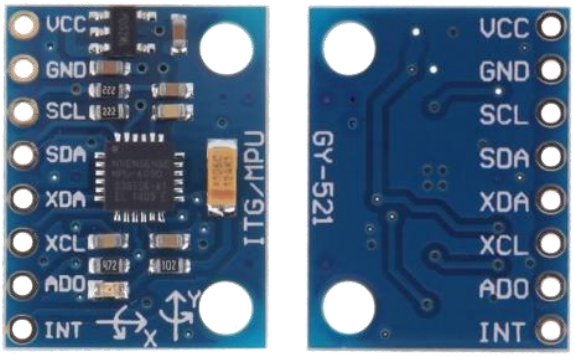
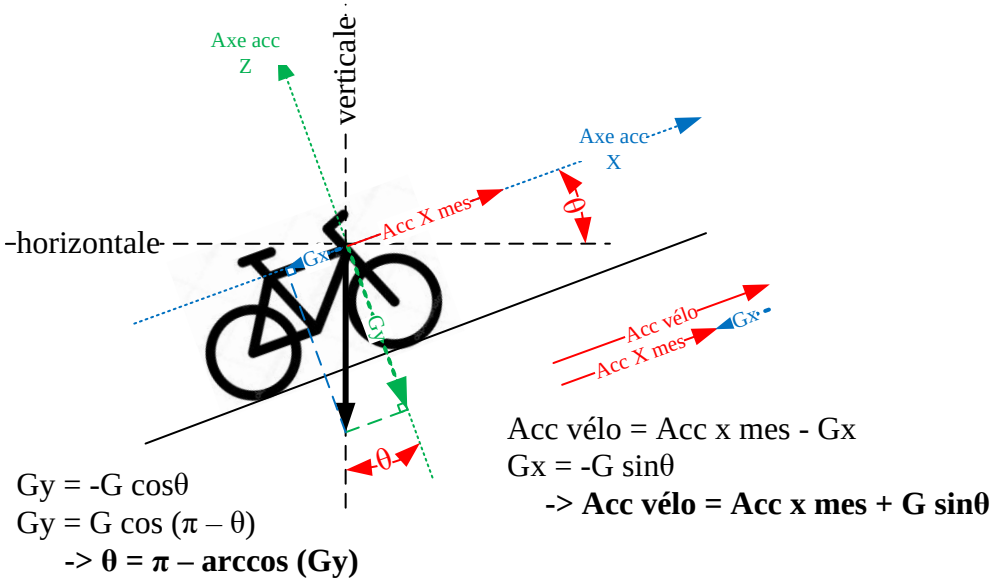


Figure 8: *Accéléromètre GY-521.*

L'accélération mesurée par l'axe X du capteur fixé sur le vélo, correspond à l'accélération du vélo si celui-ci roule à l'horizontale. Dès lors que le vélo est en montée ou en descente, le capteur mesure également une partie de la gravité terrestre sur son axe X (voir figure ci-dessous). En conséquence la valeur mesurée sur X ne coïncide plus avec l'accélération du vélo.

On montre que l'accélération du vélo dépend du sinus de l'angle d'inclinaison θ. Cet angle est déterminé grâce à la mesure d'accélération sur l'axe Z.



Gy = -G cosθ
Gy = G cos (π – θ)
-> θ = π – arccos (Gy)

Acc vélo = Acc x mes - Gx
Gx = -G sinθ
-> Acc vélo = Acc x mes + G sinθ

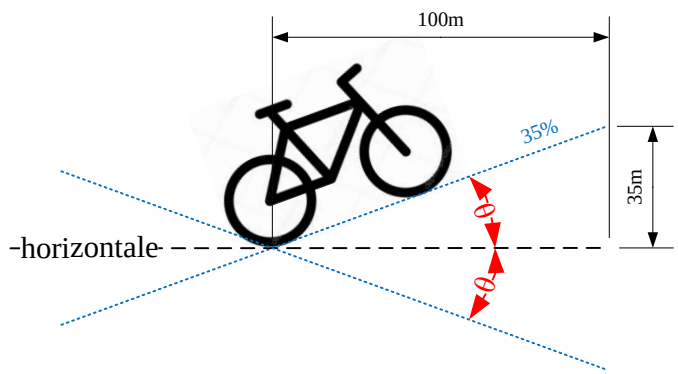
Figure 9: *Principe de mesure de l'accélération du vélo.*

Pour obtenir des résultats probants, nous avons dû mesurer, filtrer puis corriger les accélérations issues du capteur sur les axes X et Z. En effet, les signaux bruts sont trop bruités pour être directement exploitables.

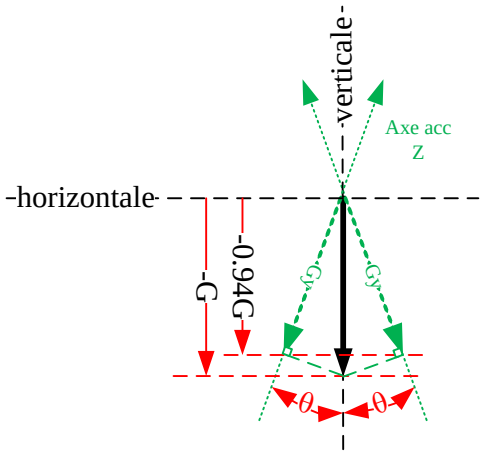
Nous avons donc appliqué, dans un premier temps, une moyenne glissante. Celle-ci est calculée perpétuellement, dès lors qu'une nouvelle donnée d'accélération est mesurée, venant remplacer la plus ancienne. Elle permet donc de lisser le signal et d'atténuer les valeurs extrêmes.

Suite à cela, nous avons opéré des corrections d'offset et de facteur d'échelle. Elles permettent respectivement d'annuler l'ordonnée à l'origine et de corriger le coefficient directeur du signal.

Enfin, nous avons défini des valeurs cohérentes de l'accélération sur Z pour filtrer les valeurs encore trop extrêmes. Pour effectuer cette minoration, nous avons estimé que le vélo ne gravirait jamais des pentes supérieures à 35%.



θ max = arctan(35m/100m) = 20°
θ min = -20°



Donc Acc_Z est borné entre -G et G cos(180°-20°) = -0.94
On retient Acc_Zmin = -G et Acc_Zmax = -0.8 G

Figure 10: *Schéma de minoration de l'angle d'inclinaison du vélo.*

Nous avons finalement codé l'algorithme suivant.

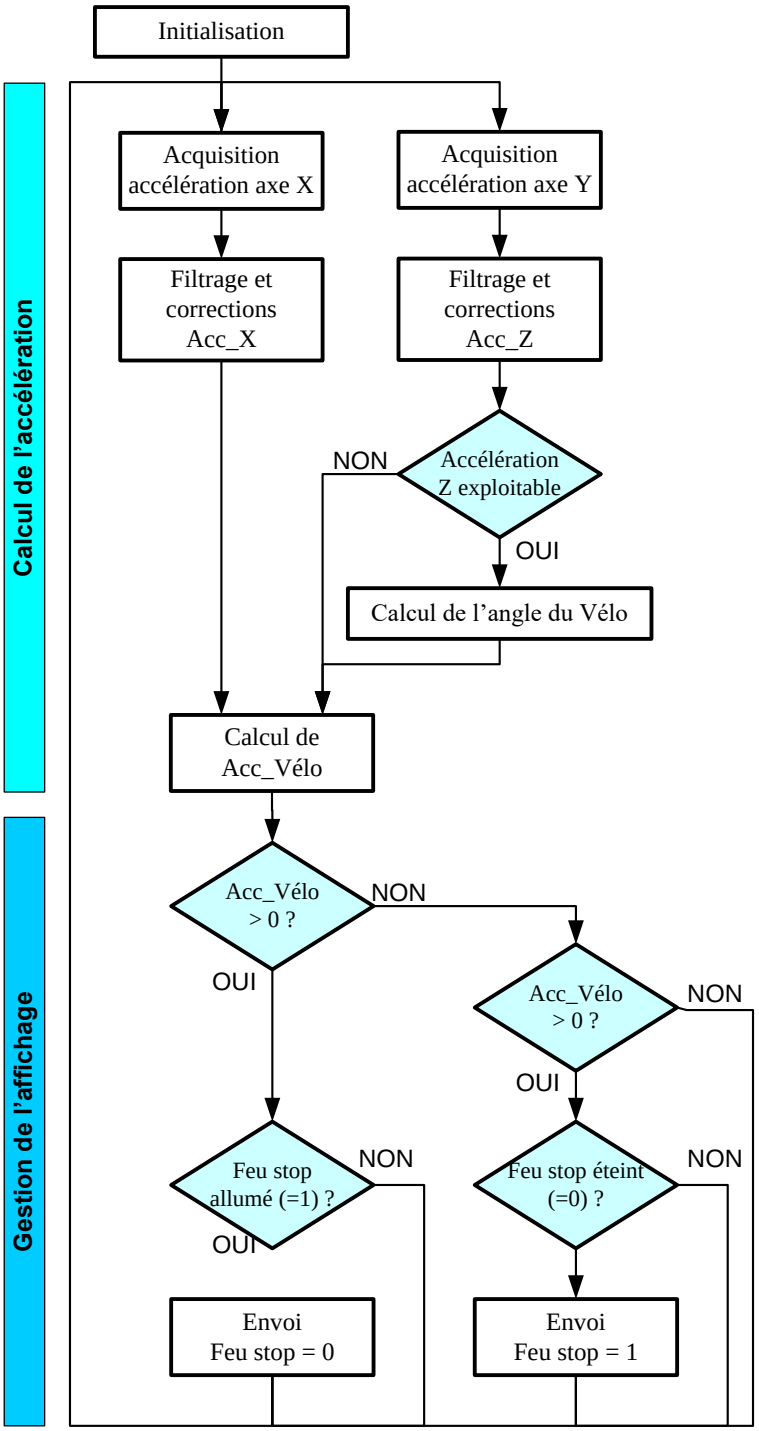


Figure 11: *Algorithme de l'acquisition et du traitement de l'accélération.*

Le calcul de l'angle d'inclinaison du vélo est plus complexe qu'il n'y paraît car il ne permet pas de déterminer si cet angle est positif ou négatif (fonction cosinus). Pour obtenir ce signe, nous utilisons le gyroscope de l'axe Y pour connaître le sens de rotation du vélo. Notre capteur étant très peu performant, nous avons décidé d'évaluer ce signe qu'une seule fois après chaque passage par la position horizontale.

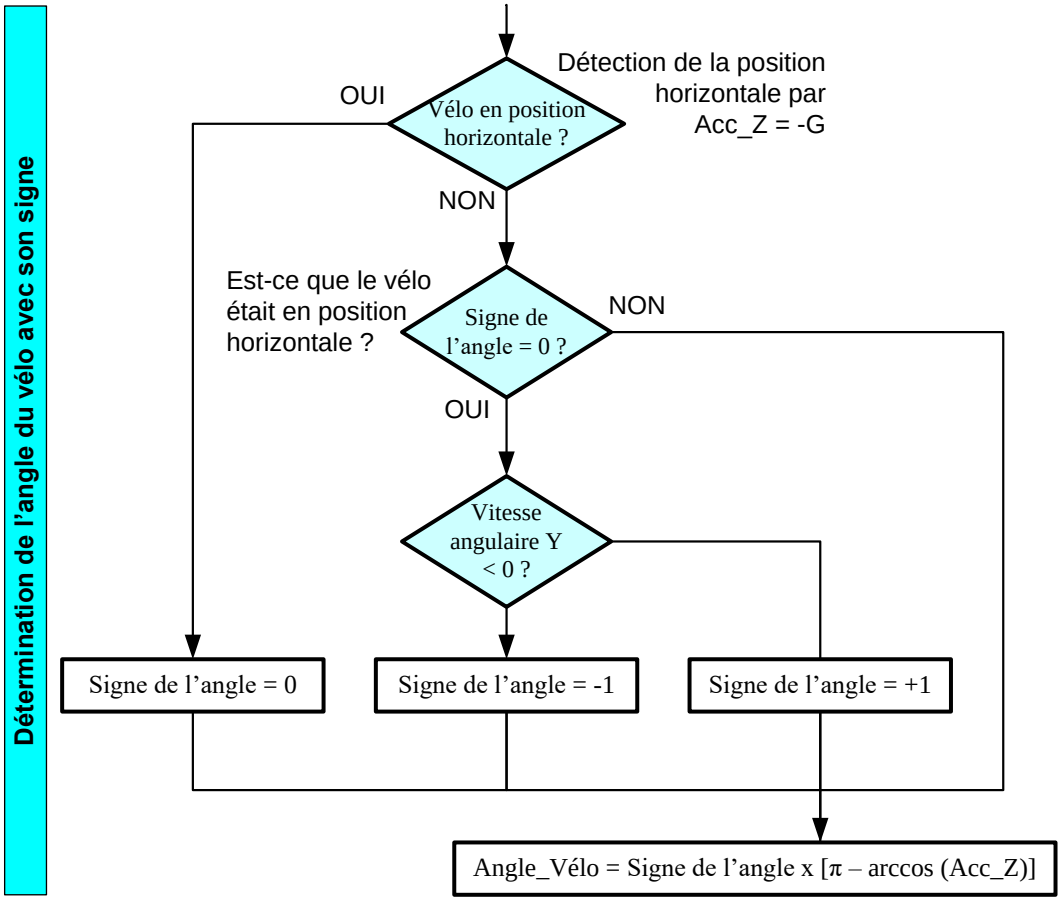


Figure 12: *Algorithme permettant de déterminer le signe de l'angle d'inclinaison du vélo.*

5.1.3 FONCTION FC6 : TRANSMETTRE LES DONNEES

Pour remplir cette fonction, nous avons utilisé deux modules Bluetooth. Le module HC-05 est utilisé en maître sur le vélo et le HC-06 en esclave sur le casque.

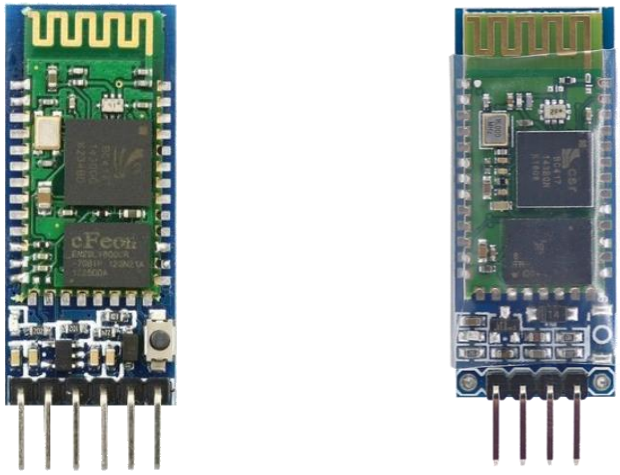


Figure 13: *Modules HC-05 et HC-06 compatible Arduino.*

Ces deux modules permettent une communication sans fil, leur portée d'une dizaine de mètres est suffisante pour notre projet et, une fois couplés, la connexion est rapide et automatique. Ce sont les raisons qui nous ont conduit à choisir ces modules.

5.1.4 FONCTION FC1 : AVERTIR L'UTILISATEUR

Afin que l'utilisateur du casque connaisse l'état des clignotants, du feu stop, de l'éclairage et du phare, nous avons ajouté trois LEDs témoins, formant un tableau de bord sur le guidon du vélo.
Une LED verte s'allume lorsque soit l'éclairage du casque est allumé (c'est-à-dire lorsqu'il fait nuit), soit lorsque le phare a été commandé par l'utilisateur.
Une LED orange clignote lorsqu'un des clignotants est allumé. Elle ne permet pas de distinguer le clignotant gauche du droit, mais uniquement d'informer l'utilisateur qu'un de ses clignotants fonctionne.
Et enfin, une LED rouge s'illumine lorsque le vélo décélère, en même temps que le feu stop s'allume sur le casque.

5.2 LA MISE EN FORME

Après avoir étudié individuellement chaque fonction du contrôle de casque, nous avons réalisé la mise en forme physique. Pour cela, nous avons branché les différents modules à une carte Arduino Nano afin de réduire au maximum l'espace utilisé. Nous avons placé l'ensemble dans un boîtier conçu sur le logiciel SolidWorks, en faisant en sorte de pouvoir le fixer sur le guidon. Ce boîtier a été imprimé en 3D ce qui permet de vérifier qu'il répond bien aux besoins d'embarquer l'électronique et de la protéger.

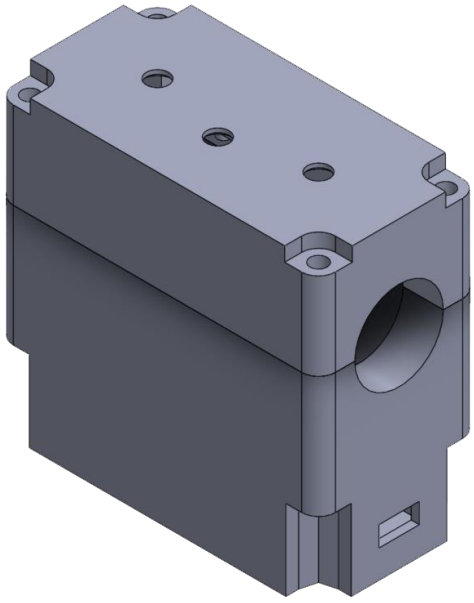


Figure 14: *Assemblage du boîtier sur SolidWorks.*

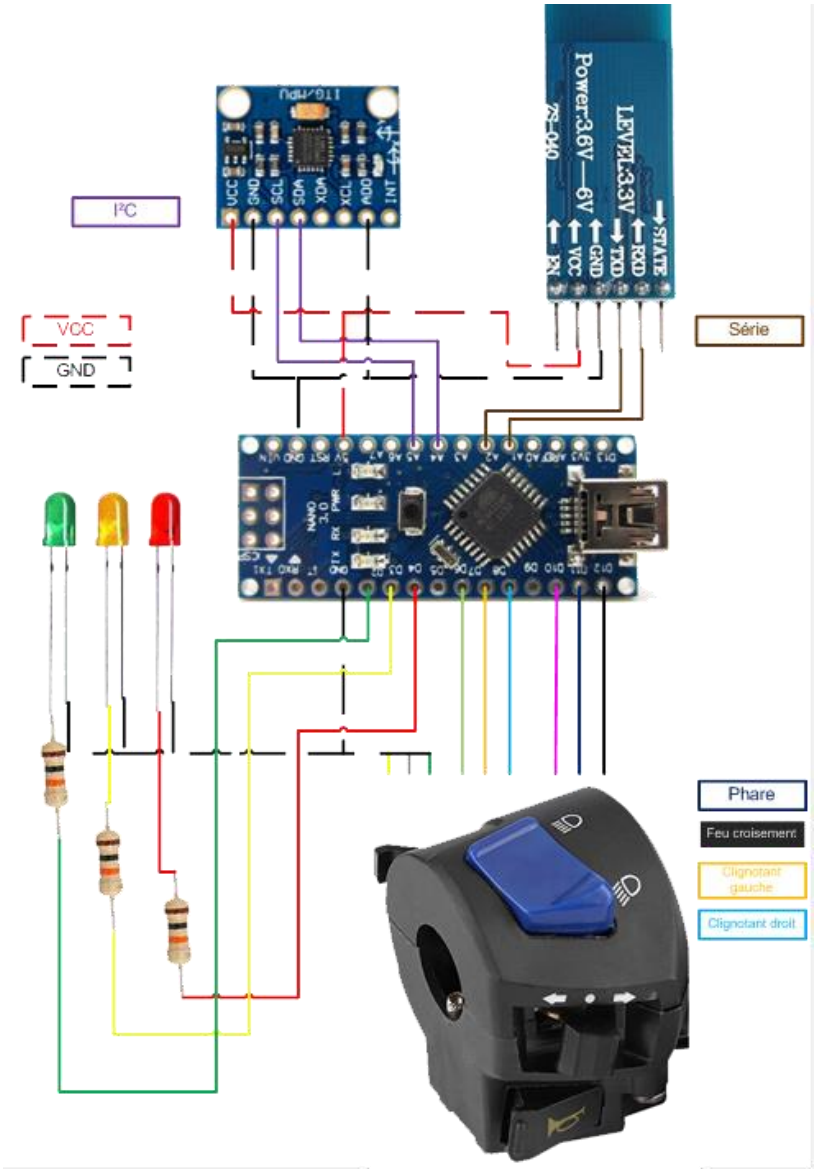


Figure 15: *Schéma des branchements des commandes.*

5.3 LES PROBLEMES RENCONTRES

Malgré les filtrages et les corrections que nous avons appliqués à la détection et au traitement des données d'accélération, la détection du freinage ne respecte pas les besoins définis dans le cahier des charges. Actuellement, la solution de l'accéléromètre ne permet pas à l'utilisateur de voyager en sécurité.

En effet, les feux stop ne s'allument que lorsque le cycliste freine très fortement. De plus, notre filtrage contre les chocs verticaux ne semble pas être assez efficace pour éviter le clignotement des feux stop. Cependant, nous avons réussi à prendre en compte l'inclinaison du vélo pour le calcul de son accélération.

Nous supposons que le problème réside dans la gamme de l'accéléromètre GY-521. Effectivement, le module est actuellement utilisé avec sa gamme minimale, à savoir ± 2g. Seulement, après plusieurs recherches, nous avons établi que l'accélération moyenne d'une voiture en ville se situait aux alentours de 0.3g. On peut donc facilement en déduire que l'accélération moyenne de notre cycliste ne dépassera jamais cette valeur. De ce fait, nous n'utilisons pas plus de 15% de la dynamique du GY-521 lors de nos mesures. Cela ne nous permet pas d'être assez précis pour notre utilisation.

La solution serait donc de changer d'accéléromètre en veillant à ce que la gamme soit adaptée et qu'il possède un gyroscope à 3 axes. Sinon, il faudrait abandonner l'idée et trouver un moyen de détecter directement l'activation des manettes de frein par le cycliste.

6 L'AFFICHAGE SUR LE CASQUE

6.1 LES DIFFERENTES FONCTIONS

6.1.1 FONCTION FC4 : SIGNALER SA PRESENCE LA NUIT

Pour notre projet, il est nécessaire de différencier le jour de la nuit car l'affichage de certaines fonctions en dépend. C'est pour cela que nous avons ajouté la fonction « Signaler sa présence la nuit ». En effet, les rappels à l'avant, le phare et une bande témoin à l'arrière doivent s'allumer lorsqu'il fait sombre.

Pour cela, nous avons utilisé une photorésistance que nous avons placée sur le casque.

Figure 16: *Photorésistance*

La valeur ohmique de la photorésistance varie en fonction de la luminosité ambiante. En insérant cette photorésistance dans un pont diviseur de tension, on peut déterminer la luminosité au travers de la tension à ses bornes. Cette tension est mesurée grâce à une entrée analogique de la carte Arduino. Nous avons ainsi pu convertir cette tension en un nombre entre 0 et 1023. Ensuite nous avons utilisé une fonction à deux seuils, appelée hystérésis, afin d'éviter les clignotements à la pénombre. Finalement, nous envoyons les caractères 'n' et 'j' à l'HC-05 suivant s'il fait nuit ou jour pour allumer le témoin sur le tableau de bord.

Voici à quoi s'apparente cette fonction.

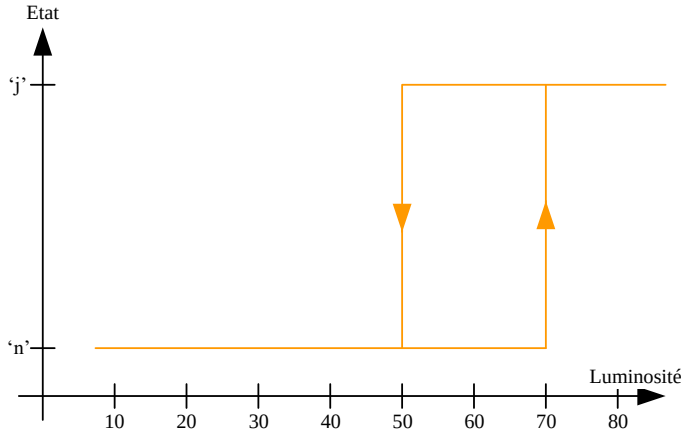


Figure 17: *Graphique de l'hystérésis*

Notre choix s'est porté sur une photorésistance plutôt qu'une horloge car la détection de la luminosité est plus pratique, dans notre cas, que la détection de l'heure. En effet, si l'utilisateur passe, par exemple, dans un tunnel, la bande témoin arrière, le phare et les rappels à l'avant doivent s'allumer même si nous sommes en pleine journée.

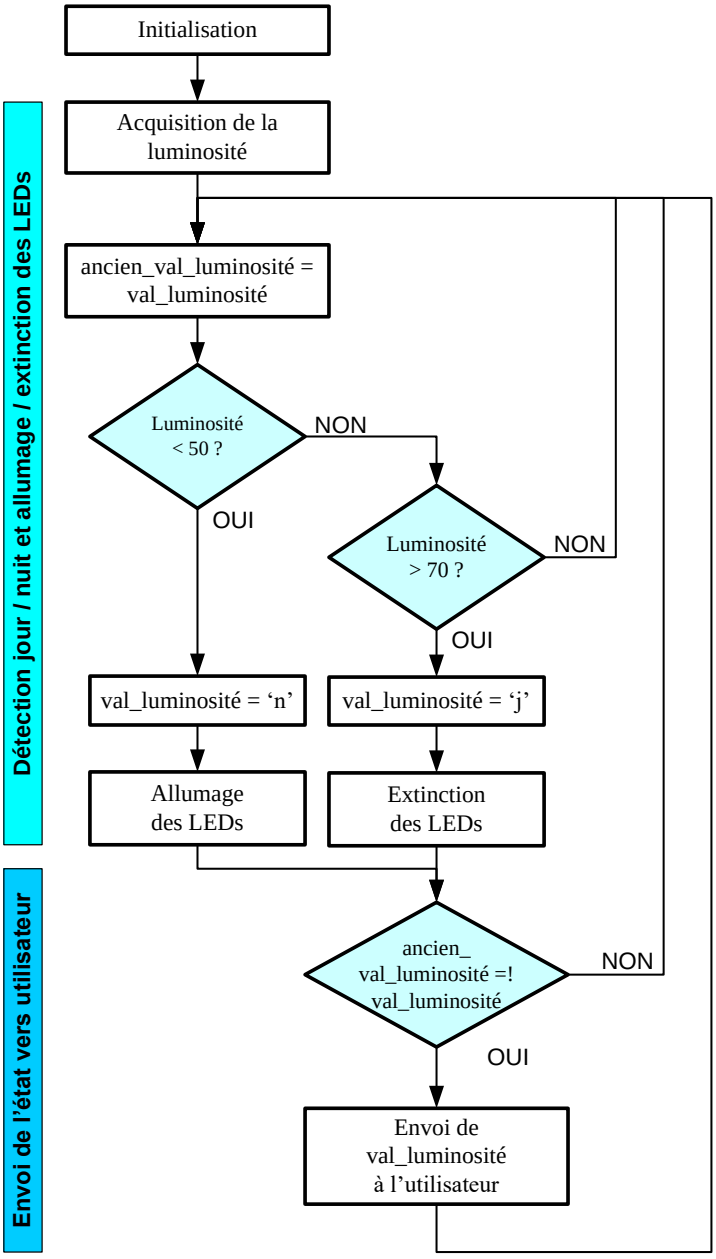


Figure 18: *Algorithme de la mesure et du traitement de la luminosité.*

6.1.2 FONCTION FC5 : ECLAIRER LA ROUTE

Pour la fonction « Eclairer la route », nous avons simplement disposé une LED de puissance à l'avant du casque en guise de phare. Celle-ci s'allume et s'éteint automatiquement lorsqu'il fait jour ou nuit et peut être contrôlée directement à l'aide d'un interrupteur sur le guidon du vélo.

Figure 19: *LED de puissance.*

6.1.3 FONCTION FC7 : SIGNALER UN CHANGEMENT DE DIRECTION

Afin de remplir la fonction « Signaler un changement de direction », nous avons décidé de mettre des clignotants sur le casque. Nous les avons placés de manière à ce qu'ils soient visibles à l'arrière et sur les côtés. Nous avons également ajouté des rappels à l'avant. Leurs fonctionnements sont directement reliés aux commandes se situant sur le guidon de l'utilisateur. Lorsque le module Bluetooth HC-06 reçoit les caractères 'g' ou 'd', cela signifie, respectivement, que le clignotant gauche ou le droit doit être allumé. Lorsque le module Bluetooth reçoit le caractère 'e', les deux clignotants doivent être éteints.

Pour ce faire, nous avons utilisé des bandes LEDs que nous faisons clignoter en orange lorsqu'un des clignotants est activé. Pour les utiliser, nous avons dû inclure la bibliothèque « NeoPixel » à notre programme.

Nous avons choisi cette solution car toutes les LEDs de chaque bande sont indicées, ce qui permet de les allumer indépendamment les unes des autres. De plus, nous pouvons choisir la couleur pour chacune des LEDs. Enfin, leur côté adhésif permet une intégration facile au casque.

Figure 20: *Ruban LEDs utilisé.*

6.1.4 FONCTION FC8 : SIGNALER UN FREINAGE

Pour la fonction « Signaler un freinage », nous avons utilisé ce même ruban LEDs afin de concevoir des feux stop situés à l'arrière du casque. Ceux-ci s'allument lorsque le module Bluetooth HC-06 reçoit le charactère 'f' et s'éteignent avec le message 'a'.

Afin de minimiser le matériel embarqué sur le casque, nous avons décidé d'utiliser les mêmes bandes LEDs pour les clignotants et les feux stop. De ce fait, lorsque l'utilisateur freine et qu'un des clignotants est enclenché, les bandes LEDs concernées alternent entre la couleur rouge et orange.

6.2 LA MISE EN FORME

De la même façon que pour la partie 'contrôle du caque' du projet, nous avons réuni les fonctions, traitées jusqu'alors séparément, pour les assembler. Nous avons branché tous les composants à une carte Arduino Mega et réalisé les feux stop, les clignotants, le phare et la bande arrière témoin. Une fois les branchements effectués, nous avons fixé le tout sur le casque à l'aide de Patafix, par faute de temps.

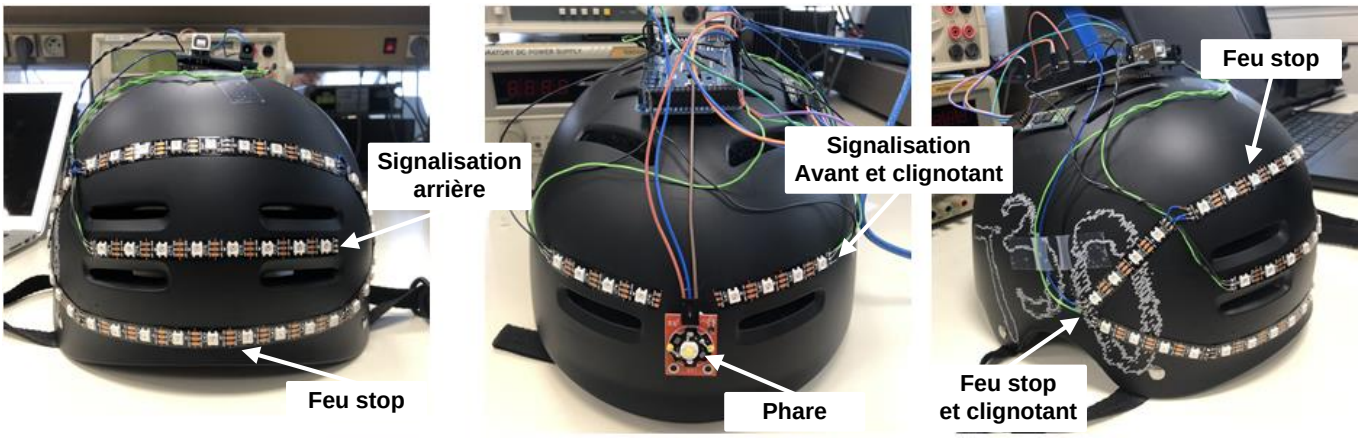


Figure 21: *Modélisation finale du casque de jour.*

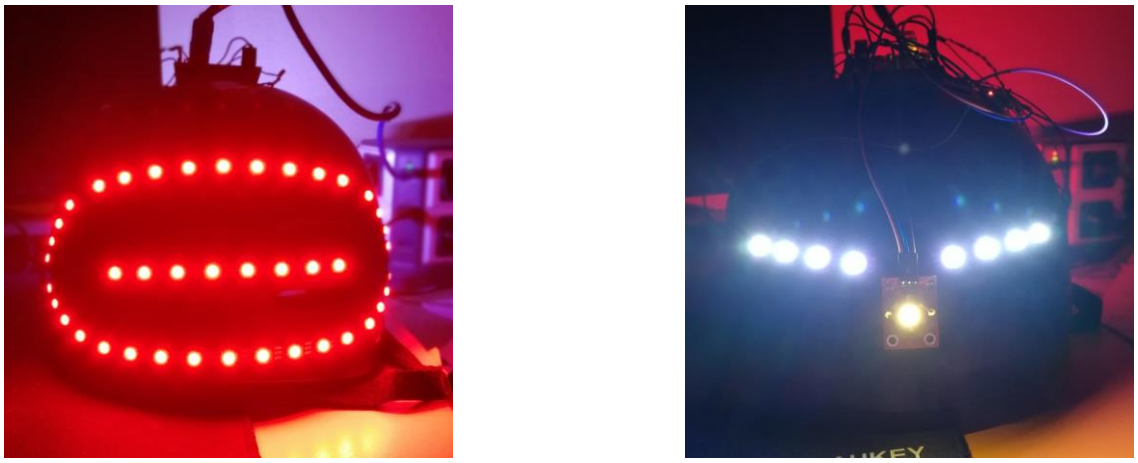


Figure 22: *Aperçu final du casque de nuit et en freinant.*

	Signalisation arrière	**Signalisation / clignotant avant**	**Clignotant arrière**	**Feu stop**	**Phare**
Freinage	-	-	-	Rouge permanent	-
Nuit	Rouge permanent	Blanc permanent	-	-	Blanc permanent
Clignotant	-	Orange intermittent	Orange intermittent	-	-
Clignotant et freinage	-	Orange intermittent	Orange / Rouge intermittent	Rouge permanent	-
Clignotant de nuit	Rouge permanent	Orange / blanc intermittent	Orange intermittent	-	Blanc permanent

Figure 23: *Etat des LEDs en fonction de la luminosité et de l'évolution des déplacements.*

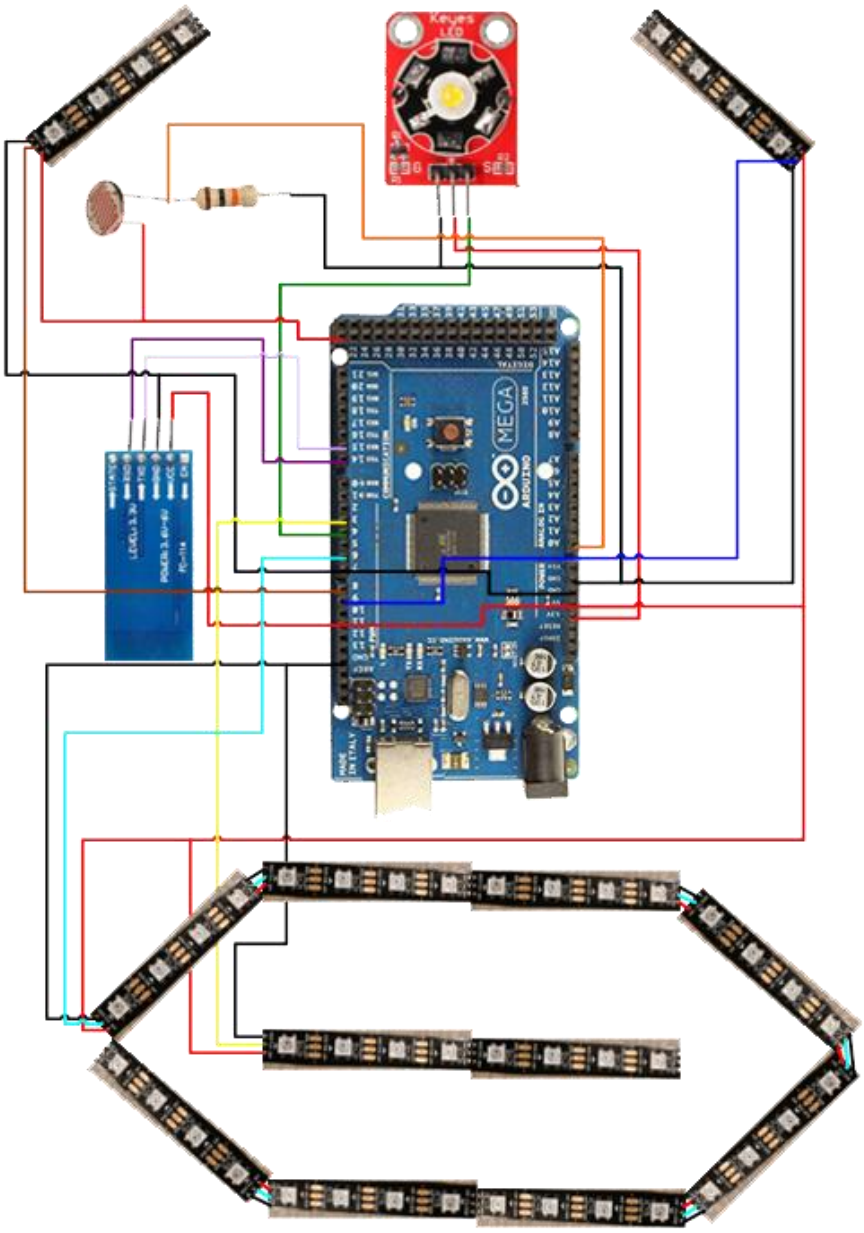


Figure 24: *Schéma des branchements du casque.*

6.3 LES PROBLEMES RENCONTRES

Lorsque nous avons regroupé toutes les fonctions en un seul programme, nous avons eu des problèmes concernant la transmission des données. Effectivement, au lieu de recevoir les caractères convenus, nous réceptionnions aléatoirement des points d'interrogations ' ?'. Nous avons alors pensé que la consommation en courant des bandes LEDs était trop importante et perturbait la communication entre les deux modules Bluetooth. Cependant, après avoir essayé de séparer les deux fonctions avec deux alimentations différentes, le problème persistait.

Finalement, il s'est avéré qu'en utilisant une carte Arduino Mega plutôt qu'une carte Uno, le problème fut résolu. Nous pensons donc qu'il y avait une interférence entre les bibliothèques « NeoPixel » et « SoftwareSerial » servant respectivement à l'allumage des LEDs et à l'émulation de la liaison série. En effet, en utilisant une carte Arduino Mega, nous n'avions plus besoin d'importer la bibliothèque « SoftwareSerial » car cette carte dispose de plusieurs liaisons séries physiques.

7 CONCLUSION

Au travers de notre casque intelligent, nous avons répondu en grande partie à notre problématique :

Comment améliorer la visibilité des cyclistes ?

Nous avons adapté les équipements et la signalétique d'un véhicule motorisé à un casque de vélo. Celui-ci comporte finalement un phare, des clignotants gauches et droits visibles à 360° autour du cycliste et d'une bande de signalisation de nuit avant et arrière. Il s'accompagne d'un tableau de bord et de commandes sur le guidon qui communiquent sans fil avec le casque.

Si nous devions poursuivre ce projet, nous reviendrions sur la détection et le traitement de l'accélération, en remplaçant l'accéléromètre, afin que la fonction « Signaler un freinage » soit plus performante. Nous travaillerions sur l'alimentation des deux cartes Arduino et nous soignerions la mise en forme finale du casque. En effet, celui-ci n'est pas vraiment utilisable en conditions réelles : il s'agit d'un prototype. Pour cela, nous concevrions un boîtier, représenté sur la figure ci-dessous, afin d'embarquer et de protéger toute l'électronique présente sur le casque.

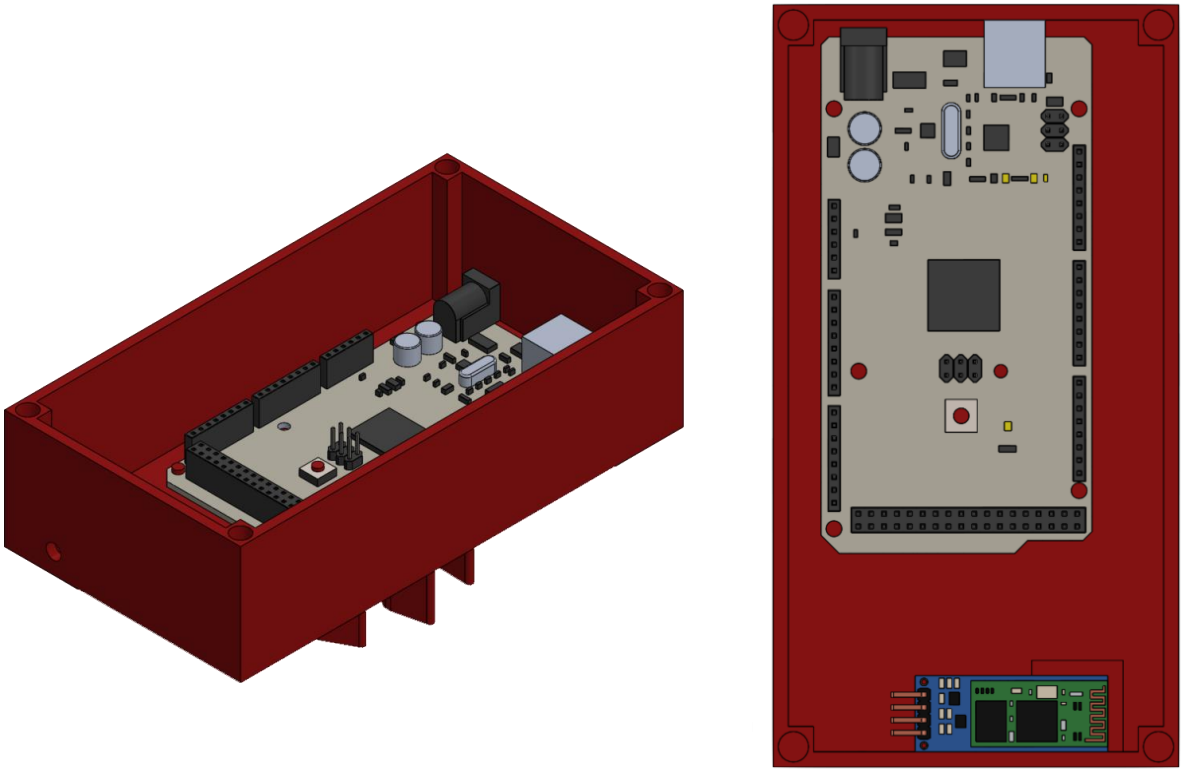


Figure 25: *Forme de boîtier pour le casque.*

8 BIBLIOGRAPHIE

Moyenne glissante

http://www.educatim.fr/tq/co/Module_TQ_web/co/moyenne_glissante.html

Accéléromètre GY-521

https://playground.arduino.cc/Main/MPU-6050

https://www.hackster.io/Nicholas_N/how-to-use-the-accelerometer-gyroscope-gy-521-6dfc19

https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf

http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Accelerometre%20-%20Projection%20-%20MASSON.pdf

HC-06 & HC-05

http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Communications%20RF%20-%20Projection%20-%20MASSON.pdf

https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.MaterielMega2560

LED 3W

https://forum.arduino.cc/index.php?topic=505584.0

Ruban LED

https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-installation
https://learn.adafruit.com/adafruit-neopixel-uberguide

«Test_10_LED_MEGA.ino» de Mr. MASSON

Photoresistance

https://www.carnetdumaker.net/articles/mesurer-la-luminosite-ambiante-avec-une-photoresistance-et-une-carte-arduino-genuino/