Compte Rendu

Contrôle de vitesse d'un moteur à CC en utilisant un LIDAR

M1. Master en Systèmes Embarqués et Systèmes Intégrés

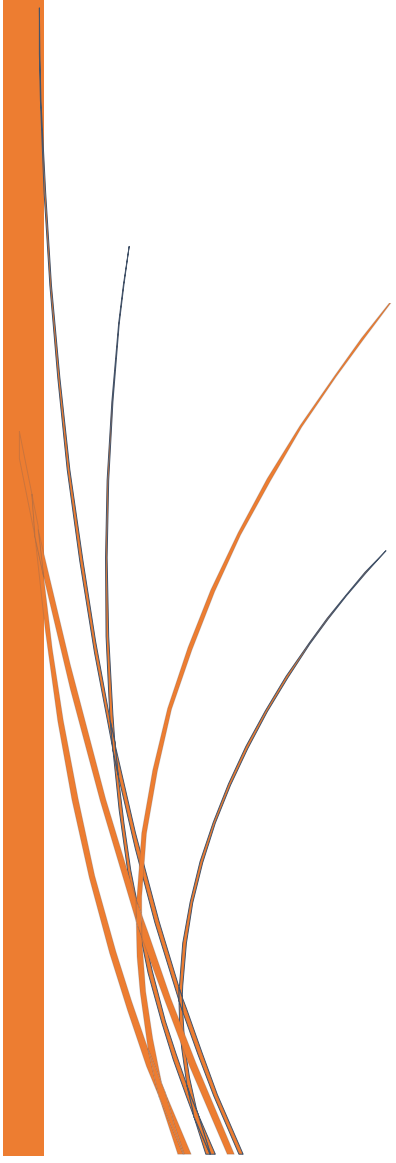
Projet Master 1

Étudiant : JAVIER LÓPEZ José Antonio

No. Etudiant : 21804920

Date de livraison :19 décembre 2018

Professeur : LAURENT Johann



Faculté sciences & sciences de l'ingénieur

Université Bretagne Sud

1. Introduction

L'objectif de ce document est d'expliquer comment la conception et la mise en œuvre du projet ont été réalisées.

2. Développement

2.1. Analyse du système

Le flux simple que le projet fait (sans boucle) est le suivant :

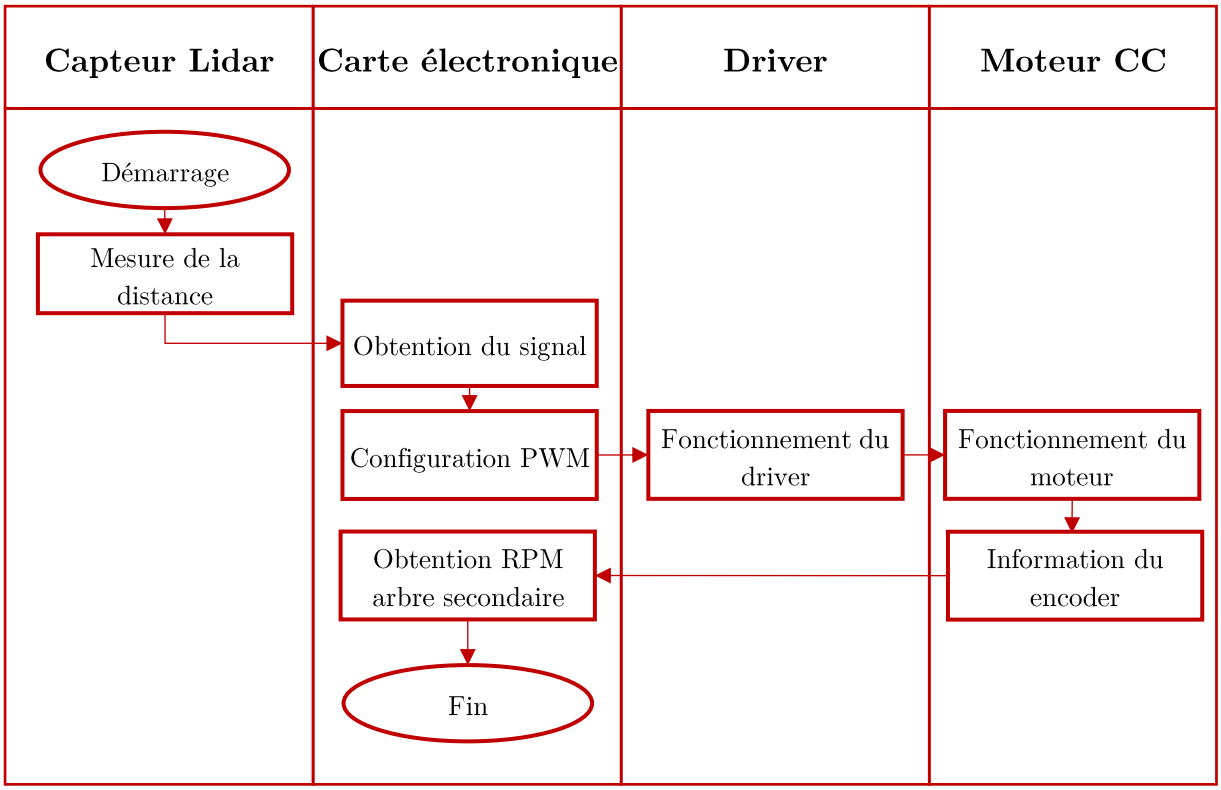


Schéma 1. Algorithme de fonctionnement

Le nombre de pins à utiliser a été analysé afin de concevoir et de créer un circuit capable de communiquer avec tous les éléments. Les composants devant communiquer sont le capteur Lidar, le driver et le moteur de courant continu. Les connexions à effectuer sont donc :

Tableau 1. Connexion inter-système

Capteur/LIDAR	Connexion	Driver	Connexion	Moteur	Connexion
Rouge(Vcc)	Vcc -STM32	**Ina**	Vcc-STM32	**Rouge**	Outa/Driver
Jaune(Activation)	PA0 STM32	**Pwm**	PA8 STM32	**Noire**	Outb/Driver
Vert(SCL)	-	**Inb**	Masse- STM32	**Vert**	Masse- STM32
Bleu(SDA)	-	**Vdd**	Vcc- STM32	**Bleu**	Vcc- STM32
Noir(Masse)	Masse-STM32	**Gnd**	Masse STM32	**Jaune**	PA6 STM32
		Outa	Rouge/moteur	**Blanche**	PA7 STM32

Capteur/LIDAR	Connexion	Driver	Connexion	Moteur	Connexion
		Outb	Noir/moteur		
		Gnd	Gnd pile		
		Vin	Vcc pile		

Vcc est la tension fournie par la carte électronique STM32, la masse est partagée entre tous les composants. En prenant en considération tous les éléments, le circuit développé est donc :

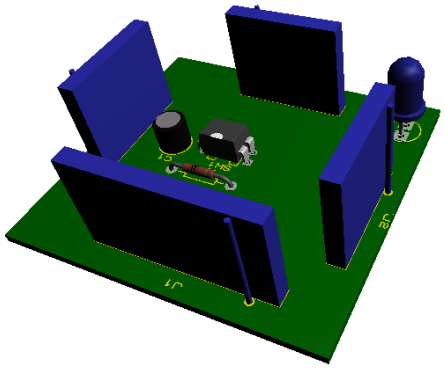


Figure 1. Conception 3D du circuit

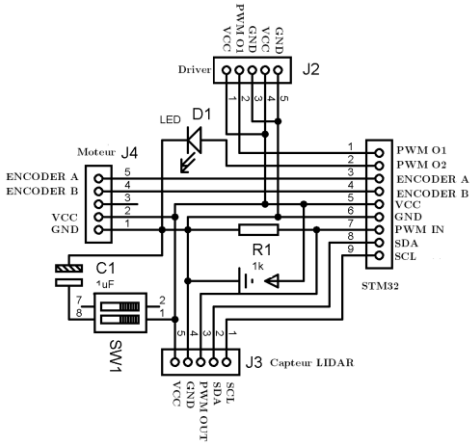


Figure 2. Conception 2D du circuit

Nous avons ajouté une LED afin de faire des tests avant d'intégrer la carte et le driver, nous avons mis aussi un switch afin de choisir entre la communication I2C ou PWM.

2.2. Acquisition de données

Le capteur LIDAR peut être géré par une communication PWM ou I2C. Les deux communications ont été tentées, mais la communication avec I2C n'a pu être établie, puisque nous avons fait une mauvaise gestion des adresses. Une communication PWM a été donc utilisée pour le développement du projet. Pour cette communication les fils vert, bleu et orange ne sont pas utilisés, les fils restants (rouge, jaune et noir) auront donc la distribution suivante :

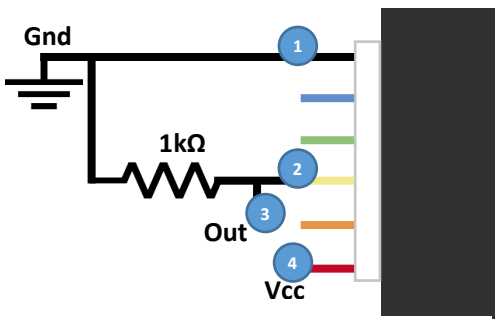


Figure 3. Connections du capteur PWM

Tableau 2. Connections du capteur PWM

#	Fonctionnalités
1	Connections à la masse
2	Connection en mode lecture continue
3	Sortie du capteur (format PWM)
4	Alimentation du capteur

Le capteur a une sortie où la largeur d'impulsion est proportionnelle à la distance mesurée. Selon sa spécification une longueur de 10 μs correspond à 1 cm. Si le capteur peut mesurer une distance maximale de 40 m, la largeur du signal est donc de 40,000 µs (0.04 s). La configuration d'un "Timer" en mode « *PWM capture* » sera donc nécessaire afin de détecter le signal attendu. Afin de réaliser la configuration du "Timer", les valeurs "Prescaler" et "Counter Period" devront être obtenues, la formule suivante sera donc utilisée :

$$T = \frac{(Prescaler + 1)(Counter\ Period + 1)}{F_{clock}}$$

Si $T = 0.04\ s, F_{clock} = 90MHz$ et nous proposons un "Period" égal à 40,000. La valeur de "Prescaler" est :

$$Prescaler = \frac{T\ F_{clock}}{(Counter\ Period + 1)} - 1 = 89$$

Pour mesurer la fréquence et la longueur d'un signal dans un état haut, la fonction "INPUT CAPTURE MODE CALLBACK" a été utilisée, puisqu'elle nous permet de surveiller le "Timer 2" et le "Channel 1'. Une condition if nous permet de vérifier l'état du "PIN A0", si elle reçoit une valeur, nous utilisons la fonction "TIM2 -> CCER &= ~TIM_CCER_CC1P" avec le but d'utiliser le "Capture Mode", lorsque le signal d'entrée est dans un état bas. Quand il y a un changement d'état (Haut->Bas) le temps est mesuré, avec la fonction "HAL_TIM_ReadCapturedValue", puis la valeur est divisée par 10 afin d'avoir la valeur de la distance en cm et nous faisons le réglage pour utiliser le "Capture Mode", lorsque le signal d'entrée est dans un état haut.

2.3. Génération d'un PWM

Dans le but de contrôler la vitesse d'un moteur à courant continu à travers la distance mesurée, la carte électronique devrait moduler la quantité d'énergie envoyée au moteur. L'utilisation d'un PWM sera donc nécessaire. Cependant les valeurs reçues par le Driver devront être de 0 à 20khz puisqu'il ne peut gérer que ces types de fréquences. Selon les tests effectués sur une LED, nous pouvons constater qu'une fréquence de 50 Hz provoque une variation continue de tension. Pour programmer telle fréquence nous devons déterminer le "Prescaler" et le "Counter Period", la formule suivante est donc utilisée :

$$T = \frac{(Prescaler + 1)(Counter\ Period + 1)}{F_{clock}}$$

Si nous connaissons la valeur de F_{PWM} , la valeur de F_{clock} et que nous proposons un "Counter Period" égal à 1799, la valeur de "Prescaler" est donc :

$$Prescaler = \frac{T\ F_{clock}}{(Counter\ Period + 1)} - 1 = 999$$

Lorsqu'une distance de 40m est mesurée, le rapport cyclique sera égal à 100%. Cependant, pour effectuer des tests dans le laboratoire, une variable *(DistanceMax)* a été ajoutée ayant le but de fixer une distance maximale et d'avoir un rapport cyclique de 100%. La formule suivante a été utilisée pour déterminer la valeur du PWM :

$$ValeurPWM = \frac{1800 * Distance}{DistanceMax}$$

La variable Distance est égale à la distance mesurée, la variable *DistanceMax* est égale à la distance choisie par l'utilisateur et 1800 est égal à le "Counter Period " programmée sur le "TIM 1". La valeur obtenue sera assignée au "Capture Compare Register (CCR)" du "TIM 1" et puisque cette valeur sera comparée avec la valeur du compteur CNT, les temps en haut et en bas seront donc modifiées.

2.4. Contrôle du moteur

Comme vu précédemment, le moteur sera contrôlé par la carte STM32, un amplificateur de puissance est alors nécessaire, puisqu'il fournira la tension et le courant au moteur. Le composant qui permettra de gérer le moteur est donc le driver VNH5019. Comme le but du projet est seulement de contrôler la vitesse, les pins qui permettent de définir le sens de rotation du moteur seront configurées pour que le moteur ne fonctionne que dans un sens. La configuration physique du driver peut être visualisée sur la Figure 2. Conception 2D du circuit. Sachant que le moteur supporte une tension de 6 à 24 volts, l'utilisation d'une pile de 9V a donc été proposé afin d'avoir un système relativement portable. Toutefois, pour effectuer des essais de charge, la source à utiliser devra être capable d'administrer un courant maximal de 2,5 ampères, la pile devra donc être remplacé.

Afin d'obtenir le régime du moteur, le "TIM3" a été configuré en "Encoder mode". Sachant que l'arbre secondaire donne 8256 impulsions afin d'avoir un tour sur l'arbre principal, les paramètres du encoder ont été mis de la façon suivante "Period" égal à zéro et "Counter Period" égal à 8256. Une quatrième horloge a été configurée afin d'avoir une interruption chaque seconde. Les valeurs de configuration utilisées avec les formules précédemment utilisées sont donc "Counter Period" égal à 8,999 et "Prescaler" égal à 9,999. Sur la fonction de l'interruption "TIM4" (stm32f7xx_it.c), nous faisons le calcul du régime moteur. Nous lançons la fonction "HAL_TIM_Encoder_Start" puis nous obtenons la valeur du compteur (égale au nombre de pulsations du encoder). Comme la valeur obtenu est proportionnelle à une seconde on multiplie la valeur par 60 afin d'avoir le régime du moteur par minute.

2.5. Intégration du système

Le système a été intégré par différentes étapes :

- La première étape a été l'intégration du capteur LIDAR avec la carte STM32 comme le but d'acquérir les données envoyées par le capteur. Dans cette étape nous avons mesuré la sortie du capteur, afin de valider la proportionnalité de 10µs égale à 1 cm.

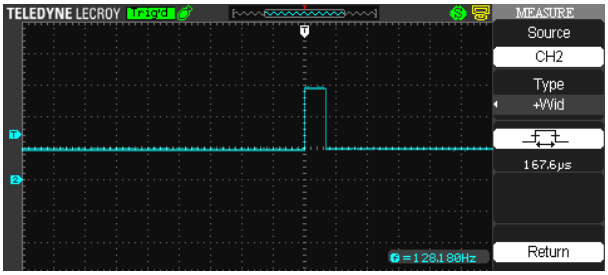


Figure 4. Sortie du Capteur 167.5 µs ≈ 17 cm

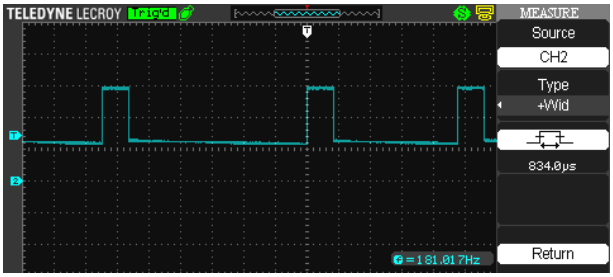


Figure 5. Sortie du Capteur 830.0 µs ≈ 83 cm

Différents tests ont été faits pour valider la distance mesurée par le capteur LIDAR versus la distance reçue par la carte électronique STM32.

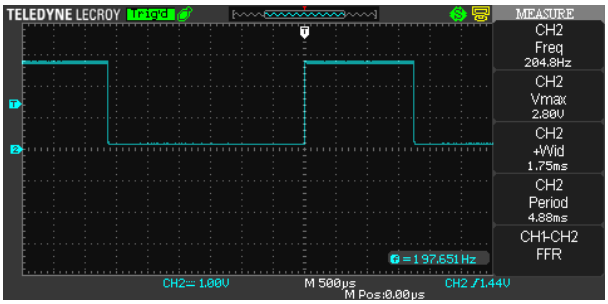


Figure 6. Signal de sortie du capteur

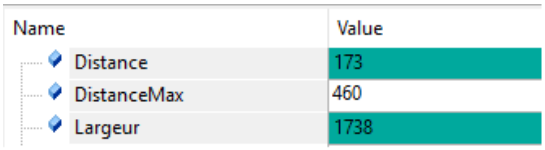


Figure 7. Distance reçue par la carte

Une certaine différence a été détecté entre les valeurs envoyés et reçues. Cette différence est provoquée parce que le capteur ne reste jamais dans un état stable, sa valeur variée entre ± 2,5 cm.

- La deuxième étape a été la configuration d'une sortie au format PWM. Une fois que le TIMER a été configurée avec les valeurs précédemment obtenues, la sortie de la carte a été analysée dans l'oscilloscope afin de valider la fréquence programmée et le fonctionnement du rapport cyclique.

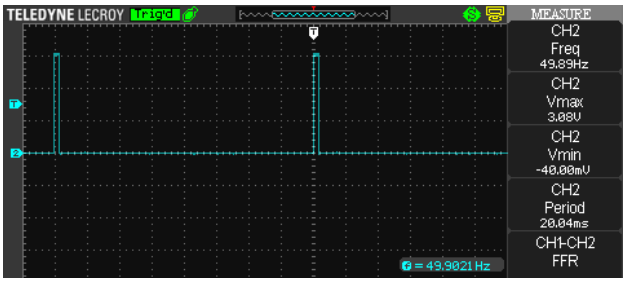


Figure 8. PWM (Rapport cyclique 2%-Fréquence 50Hz)

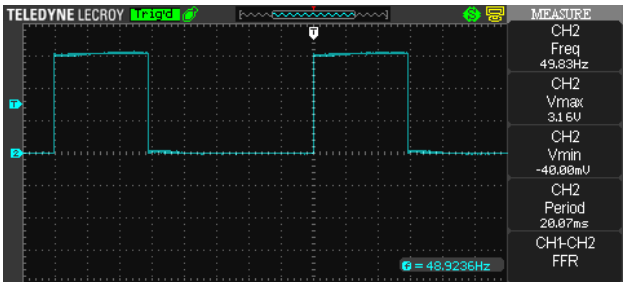


Figure 9. PWM (Rapport cyclique 37.5%-Fréquence 50Hz)

Nous pouvons constater que la fréquence programmée sur la carte est identique à celle indiquée sur l'oscilloscope, les valeurs proposées de "Preescaler" et "Counter Period" ont donc été validées.

- La troisième étape a été l'intégration du driver avec la carte STM32. Grâce à la LED présente, nous avons pu constater que le driver reçoit de façon correcte le signal PWM de la carte.

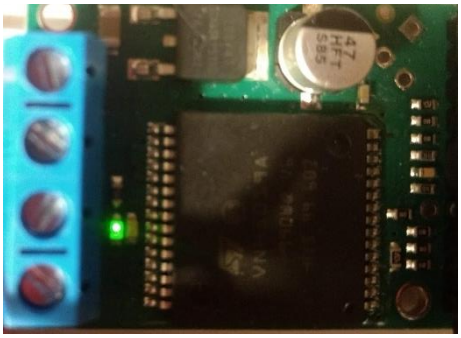


Figure 10. PWM (Rapport cyclique 2%-Fréquence 50Hz)

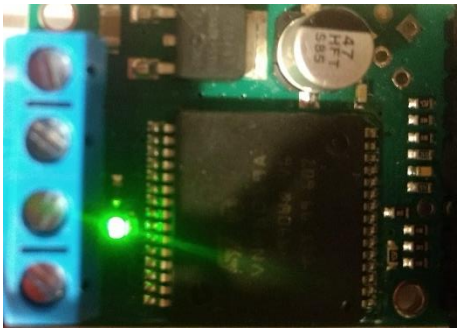


Figure 11. PWM (Rapport cyclique 37.5%-Fréquence 50Hz)

- Le moteur a été intégré avec la pile. Les fils d'alimentation du moteur ont été connectés à la sortie du Driver, et les fils de l'encodeur ont été connectés à la carte STM32. L'ensemble du système a été testé et il fonctionne comme prévu. Lorsqu'un objet s'approche, le moteur ralentit sa vitesse. Cette configuration a été faite parce que ces types de systèmes sont utilisés dans des véhicules autonomes. La configuration tente donc d'émuler le comportement de ces véhicules.
- Finalement le circuit construit pour faire des tests sur le système a fonctionné correctement, ce qui a permis de gagner du temps sur tous les tests effectués et de s'assurer que tous les fils ont été correctement connectés.

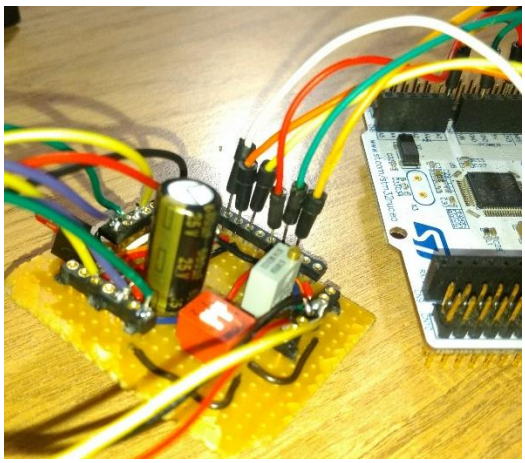


Figure 12. Circuit du système

2.6. Résultats

Afin de vérifier le changement de vitesse du moteur, une distance maximale de 200 cm a été fixée. Le changement de vitesse a été mesuré pour une distance allant de 0 à 200 cm Les résultats suivants ont été obtenus :

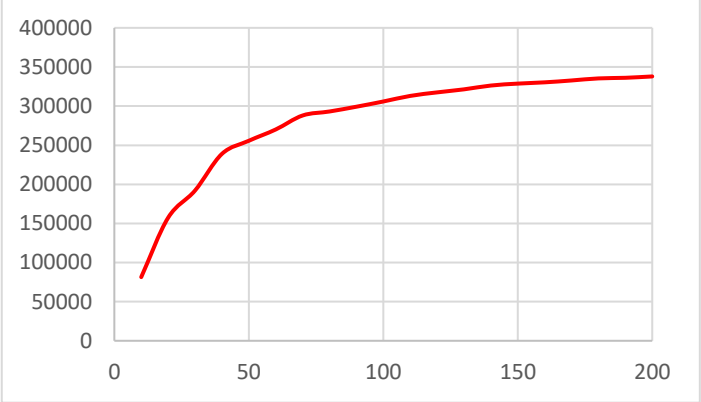


Figure 13. Validation de changement de vitesse

L'objectif principal est donc validé, le moteur varie sa vitesse en fonction de la distance mesurable. Afin de mesurer la précision du capteur, 20 échantillons ont été pris (0-200cm). La marge d'erreur moyenne obtenue est de ±1,75 cm. Les résultats sont observés dans la figure suivante :

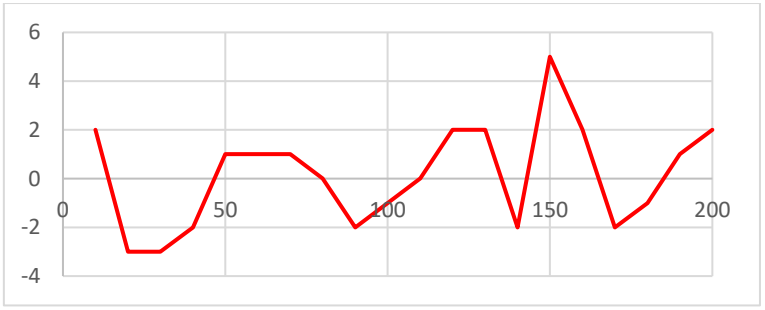


Figure 14. Marge d'erreur vs distance

Le capteur répond à ses spécifications, cependant nous avons constaté qu'il n'est jamais stable et ses valeurs changent de forme en continue entre, ± 2,5 cm. La plage de fonctionnement que nous avons trouvé est de 10 cm jusqu'à 4000 cm. Ces variations peuvent être causées par des variations de lumière et de matériaux, puisque le capteur fonctionne en envoyant et recevant des faisceaux de lumière infrarouges.

En prenant comme référence le cahier des charges, toutes les fonctions proposées ont été remplies (Récupérer des données d'un capteur qui mesure la distance afin de définir la vitesse d'un moteur à CC, utiliser une carte électronique pour recevoir et envoyer des données, réagir en fonction de l'environnement, capturer les informations reçues par le capteur, commander un moteur selon les données reçues au moyen d'un circuit de puissance). La date estimée d'achèvement du projet a été aussi correctement atteinte.

3. Conclusion

La programmation sur Keil a été vraiment difficile. En effet même si c'était un langage C, le logiciel a plusieurs fonctions qui peuvent être utilisées pour développer le projet. De plus, trouver des informations sur le sujet est un peu compliqué puisque la communauté des cartes stm32 qui utilisent le logiciel Keil est limitée par rapport à la communauté arduino. Toutefois cela peut être cohérent car la carte n'ést pas faite pour les amateurs mais pour les développeurs de projets.

Pour mesurer presque précisément, le capteur doit être correctement centré car la plus petite inclinaison peut changer la valeur de la distance (au lieu de mesurer 9 cm le capteur peut mesurer une distance de 10 cm). Le positionnement du capteur devient plus complexe quand nous voulons mesurer des distances plus grandes puisqu'un léger changement de position provoque une variation sur la lecture. Cette erreur peut peut-être être corrigée au moyen de la communication I2C. Cependant, le capteur répond correctement à ces spécifications car la marge d'erreur moyenne est de ±2,5cm pour des mesures inférieures à 5m. L'utilisation d'une pile est pratique pour effectuer des tests, cependant nous avons oublié d'utiliser un régulateur, la pile ne peut plus donc fournir le courant et la tension nécessaires avec le temps. Les valeurs de sortie peuvent donc varier d'un test à un autre, l'utilisation d'une source fixe ou un régulateur est proposée pour éliminer cette erreur possible.

La réalisation d'un planning est important parce que de cette façon, semaine après semaine, nous avons eu de façon claire les activités à effectuer ce qui nous a permis d'avoir une meilleure gestion de notre temps. Certaines activités ont eu une durée égale à celle prévue, mais d'autres ont pris plus de temps, cependant les temps se sont compensés.

Finalement le projet peut avoir des améliorations dans son fonctionnement, par exemple mettre en œuvre une loi de contrôle afin d'avoir la vitesse désirée en tout temps, utiliser la communication i2c, entre autres.

4. Références

GARMIN. (s.d.). *Lidar Lite V3 Operation Manual* . Récupéré sur Garmin: https://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf

Pololu. (s.d.). *172:1 Metal Gearmotor*. Récupéré sur Pololu: https://www.pololu.com/product/2288

Pololu. (s.d.). *VNH5019 Motor Driver Carrier*. Récupéré sur Pololu: https://www.pololu.com/product/1451

Santoro, C. (2013). *Additional Timer Functionalities*. Récupéré sur Dipartimento di Matematica e Informatica - Universita di Catania, Italy `: http://www.dmi.unict.it/~santoro/teaching/lap1/slides/TimerCCP.pdf