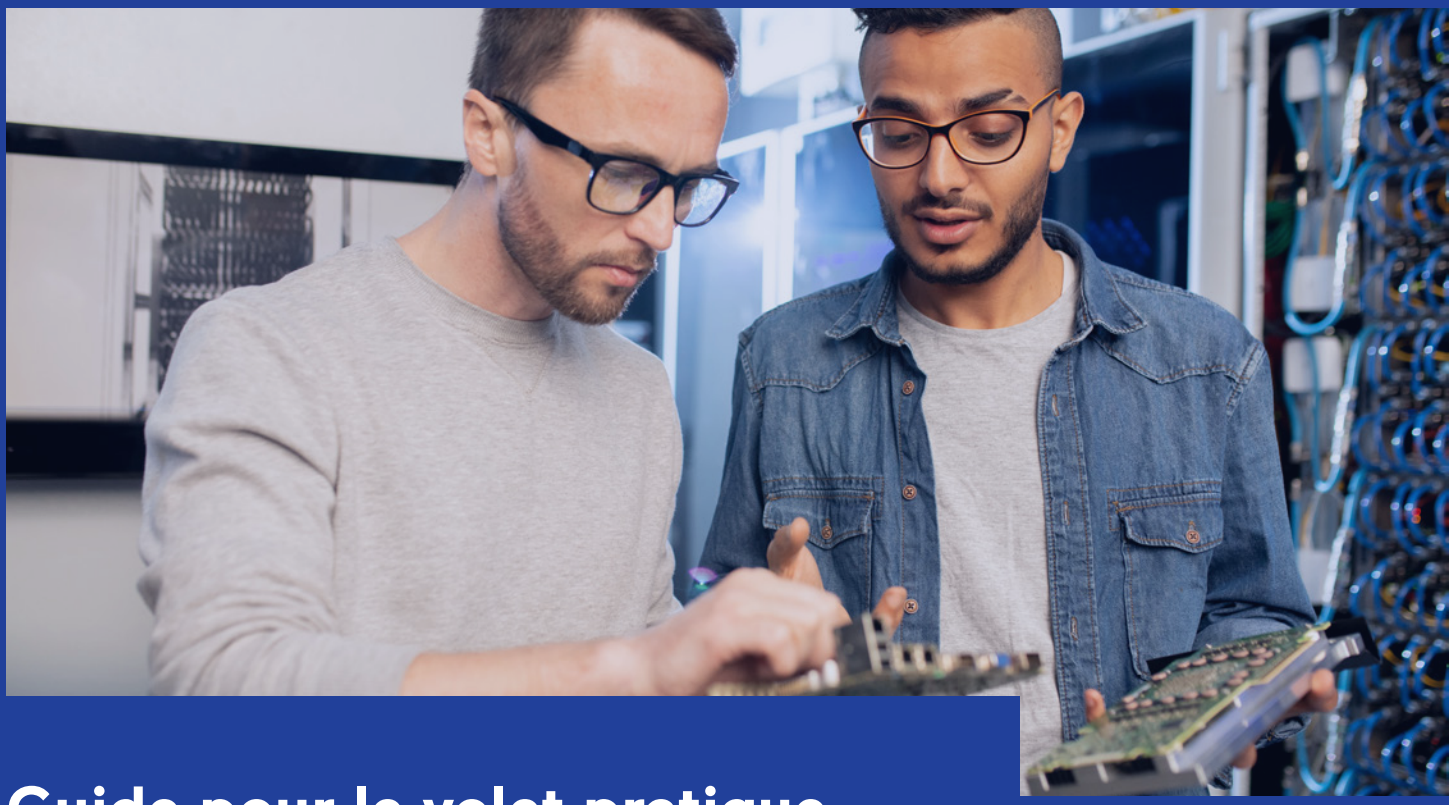
ing. Ordre des ingénieurs du Québec

Génie électrique



Guide pour le volet pratique de la formation en conception

Ce document a été préparé avec la collaboration de **Dominik Désilets, ing., M. Ing.**, chargé de cours au Département de génie des systèmes et coordonnateur des projets de fin d'études - Génie électrique à l'École de technologie supérieure

ÉTS
Le génie pour l'industrie

Table des matières

Introduction

Pour réaliser un projet en conception, on doit se baser sur la méthodologie de la conception en ingénierie dont traitent les livres suggérés dans le volet théorique. Il faut aussi recourir à des outils appropriés aux approches et aux méthodes à mettre en œuvre. C'est précisément le but de ce document : guider le candidat et la candidate dans l'accomplissement d'un projet de conception en génie électrique.

Vous trouverz dans ce guide, d'une part, des renseignements sur les outils à utiliser et les approches à appliquer et, d'autre part, un résumé des principales étapes (ou phases) d'un projet de conception en génie électrique.

En complément, nous reproduisons à titre d'exemple un projet élaboré par des étudiants.

Description du processus de conception en

génie électrique

Le génie électrique est une branche de l'ingénierie faisant appel à des connaissances et des compétences en sciences du génie, en conception d'ingénierie et en études complémentaires. L'ingénieur électrique conçoit des installations électriques, des éléments électroniques et des programmes informatiques destinés à répondre à des besoins dans différents domaines, pour des usages industriels et domestiques, à des fins de chauffage, de réfrigération, d'éclairage, de transport, de communication, de contrôle et d'alimentation de produits ou systèmes.

Les systèmes conçus par l'ingénieur électrique permettront d'élaborer des solutions pour résoudre des problèmes de natures diverses dans différentes sphères de la société. Le génie électrique touche à plusieurs domaines tels les systèmes informatiques, le contrôle industriel, les télécommunications, les systèmes embarqués, la microélectronique, la photonique et l'instrumentation médicale.

Le processus de conception repose notamment sur les connaissances acquises en mathématique, en sciences fondamentales et en sciences du génie, ainsi que sur l'application d'une méthodologie rigoureuse en tenant compte de l'environnement, de la santé et de la sécurité des utilisateurs. Il comporte trois phases : la définition du projet, la conceptualisation et la réalisation. La synthèse et le soutien technique sont généralement des éléments complémentaires à la conception.

Phase 1 – Définition du projet / analyse des besoins

La définition du projet se fait via l'analyse des besoins et mène à la rédaction d'un cahier des charges. Le but de ce cahier des charges est de clarifier les exigences et les besoins du client, de préciser le contexte dans lequel doit se réaliser le projet et de déterminer les normes applicables au projet. C'est généralement à partir d'une problématique que l'on peut définir les objectifs et les résultats attendus. Cette étape est cruciale, puisqu'elle donne les lignes directrices pour la suite du processus de conception. Une mauvaise compréhension de la problématique et des besoins des clients peut entraîner une perte de temps importante et un gaspillage des ressources.

On détermine ensuite la charge de travail et on évalue les ressources (humaines, matérielles et financières) nécessaires pour mener le projet à terme. La reconnaissance des risques et des contraintes, de même que l'émission d'hypothèses, font aussi partie de cette première phase. La planification estimée des activités permettra de mettre les bases pour le déroulement du projet en fonction des délais à respecter pour son achèvement. Il est important de spécifier la chronologie (timeline) des activités et les jalons (check point) qui serviront à vérifier l'avancement du projet, et de prévoir des périodes consacrées aux rencontres entre les différents intervenants.

Afin d'assurer une bonne communication pour le déroulement, il est impératif de déterminer les canaux de communication et les outils de visioconférence (dans un contexte de projet à distance) auxquels on aura recours.

Tableau 1 – Outils d'analyse des besoins / définition du projet

Activités	Outils
Définition des besoins clients	Rencontres, devis de projet, questionnaires
Définition des contraintes	Rencontres, devis de projet, questionnaires, remue-méninges (brainstorming), recherches sur les technologies disponibles, sur les protocoles de communication
Rédaction du ou des cahiers des charges	Word, Google Docs
Compilation de données	Excel, Access
Planification des activités	Google Calendar, Microsoft Project,
Moyens de communication	Zoom, Teams, Google Meet

Phase 2 – Conceptualisation ou élaboration du concept

La phase d'élaboration du concept vise à rechercher des idées et des pistes de solutions. Au cours de la phase de conceptualisation, une revue de littérature est nécessaire afin de faire les liens avec les concepts théoriques rattachés au projet. En fonction de la problématique et du type de projet, cette étape peut toucher à différentes disciplines du génie. La revue de littérature est effectuée à partir de différentes sources d'information telles que des dictionnaires, des ouvrages de référence techniques, des sites Internet, des articles scientifiques, des manuels (handbooks), les normes usuelles du domaine, etc.

La génération d'idées, la recherche et la créativité permettront d'élaborer et de développer différents concepts visant à combler les besoins du client et d'atteindre les objectifs. Il est important de garder à l'esprit que ces concepts doivent être basés sur les principes de l'ingénierie.

Toutes les idées retenues doivent être évaluées objectivement, de façon à choisir le concept qui répond le mieux aux besoins du client tout en respectant les contraintes établies. Une analyse des risques technologiques et économiques pour les parties prenantes, soit le client et les partenaires, doit être effectuée. Cette analyse de risques doit aussi prendre en considération la société et l'environnement, et mener à la définition de plans de contingences permettant de réduire, voire d'éliminer, les risques.

Par la suite, la méthodologie de travail qui sera utilisée pour la réalisation du concept sélectionné doit être élaborée. Cette méthodologie servira à décrire plus précisément les activités à réaliser pour atteindre les objectifs. La mise à jour du plan de travail permettra de réviser la planification de départ et de modifier les ressources, au besoin.

Les résultats de la phase de conceptualisation peuvent être des diagrammes fonctionnels, des schémas-blocs, des cartographies de circuits imprimés, etc.

Il est fortement recommandé de valider les choix et la mise à jour du plan de travail avec le client afin d'obtenir les meilleurs résultats possibles et d'assurer la satisfaction du client.

Tableau 2 – Outils de conceptualisation de projet

Activités	Outils
Recherche d'idées	Rencontre, remue-méninges (*brainstorming*), recherches sur les technologies disponibles, sur les protocoles de communication
Revue de littérature	Notes de cours, articles scientifiques, volumes de référence et manuels (*Electrical Engineering Handbook*, *Electronics Handbook*, *Electric Power Engineering Handbook*), etc.
Choix de concepts	Word, Google Docs, matrice de décisions, tableaux synthèses
Élaboration du diagramme fonctionnel	Visio, Lucidchart
Conception des schémas : électriques, électroniques	Micro-cap, Simscape, Quartus Prime, ADS, Altium Designer, etc.
Mise à jour du plan des activités	Google calendar, Microsoft project, Word
Moyens de communication	Zoom, Teams, Google Meet, (Google Apps)

Phase 3 – Réalisation : conception préliminaire et conception détaillée

La phase de réalisation consiste à développer le concept de façon précise et à faire un prototype fonctionnel de qualité. Il s'agit généralement de réaliser une partie matérielle et une partie logicielle de l'idée retenue. C'est au cours de cette phase que sont effectués les calculs détaillés par rapport aux paramètres et aux contraintes de la solution choisie.

Exemples de calculs : vitesse de propagation de signaux, dissipation de puissance, calibration de transformateur, flux de charge, analyse de circuits, paramètre d'un groupe d'alimentation, etc.

Cette portion du processus de conception comporte divers éléments : l'architecture finale, les plans, la liste des matériaux et composantes, le code informatique, les procédures de fabrication et d'assemblage, etc.

La phase de réalisation requiert fréquemment des tests en cours de route pour valider les éléments de conception. Il est donc nécessaire, dans ce cas, d'établir des procédures de tests ou de simulations : physiques, électroniques, numériques ou logicielles. Ces tests peuvent parfois être réalisés sur des bancs d'essais existants ; sinon, il faut en concevoir.

La phase de réalisation est généralement complétée lorsque les prototypes et les rapports sont livrés au client ou au promoteur. Ces livrables doivent être conformes aux attentes définies au départ et/ou ajustées en cours de projet. L'équipe de projet doit aussi être en mesure d'expliquer et de justifier les changements effectués durant les différentes étapes de la réalisation. On devrait aussi être en mesure de relever les difficultés rencontrées dans un objectif d'apprentissage et éventuellement d'amélioration continue. La gestion de l'approvisionnement en pièces et composantes tout en respectant le budget et les contraintes nécessite une rigueur de la part de tous les membres du groupe.

On pourrait considérer un projet comme incomplet tant et aussi longtemps que le client n'a pas donné son approbation quant aux résultats. Si le client n'est pas satisfait des résultats, il faut alors revoir les différentes phases du projet.

Tableau 3 – Outils de réalisation de projet

Activités	Outils
Conception des schémas électriques, électroniques	Micro-cap, Simscape, Altium
Programmation	C/C++, VHDL, Linux, Arduino, Python
Simulation	MATLAB, Simulink, Hypersim, SimPower System, LTspice, Modesim
Calculs mathématiques	Mathematica, Excel, calculatrice graphique
Mise à jour du plan des activités	Google calendar, Microsoft project, Word
Moyens de communication	Zoom, Teams, Google Meet
Essais	Générateurs de fonctions, oscilloscopes et multimètres, modules LabVolt

Complément

Il est important de valider les résultats avec le client tout au long du projet afin de s'assurer que ce dernier puisse atteindre les objectifs. Les modifications apportées en cours de projet devraient être documentées afin qu'on puisse s'y référer. Les modifications doivent tenir compte de l'ajustement des ressources (humaines, matérielles et financières).

Le travail d'équipe est aussi une caractéristique de tout projet d'ingénierie. La gestion efficace et la cohésion de l'équipe font partie des éléments importants à contrôler tout au long du projet. Cet aspect du projet, qui relève davantage des relations humaines, peut avoir des conséquences déterminantes sur le succès ou l'échec d'un projet, d'où le sérieux qui doit y être apporté.

Enfin, la communication est dans bien des cas au cœur du bon déroulement menant à la réussite du projet. Les intervenants, peu importe leurs rôles et responsabilités, doivent assurer une bonne transmission des informations, afin d'éviter les pertes de temps et les erreurs potentielles. La communication doit être bidirectionnelle et concerner tous les participants au projet.

Tableau 4 – Exemples de projets de fin d'études réalisés à l'École de technologie supérieure dans le cadre du programme de génie électrique

Projet	Thématique	Connaissances recherchées	Outils (hardware et software)
Conception d'un système de positionnement en temps réel	Télécommunication, informatique, systèmes embarqués	Programmation de systèmes embarqués, conception de protocoles de communication	Système de positionnement SR1020, carte de développement STM32
Conception et modélisation d'une centrale solaire	Énergie, simulation	Programmation MATLAB, électronique de puissance, conversion d'énergie	Simulateur OP4510, logiciel de simulation e-MEGASIM, e-FPGASIM, Hypersim. MATLAB
Conception d'un bloc d'alimentation intelligent multi-tensions	Énergie, systèmes embarqués, contrôle, microélectronique, commande	Conception de PCB, conception AC-DC, électronique de puissance, programmation Python	Convertisseur ÉBuck-Boost, actuateur linéaire, servo-moteur dynamixel, Altium Designer
Conception d'un logiciel pour FPGA pour effectuer le prétraitement d'un signal audio	Informatique	Programmation VHDL	
Conception d'un système d'acquisition de données provenant de multiples capteurs	Technologie de l'information et communication	Traitement de signaux, conception de PCB, gestion de projet	Altium Designer, Raspberry Pi, caméra infrarouge, capteur de CO_2, microphone, ordinateur Jetson TX2

Conception d'un système d'agrégation d'énergie renouvelable et de stockage	Énergie et commande industrielle	Simulation, modélisation et commande	MATLAB, cellules photovoltaïques
Conception de sondes de tensions et de courants génériques pour prises de mesures sur l'alimentation d'un prototype de robot	Informatique, systèmes embarqués	Programmation embarquée, conception de PCB, traitement de signaux, Python, ROS	Altium/Eagle, chargeur embarqué, ordinateur portable
Conception d'un outil de triangulation sonore utilisant des hydrophones avec un STM32	Télécommunication, informatique, simulation	Programmation MATLAB, Programmation C/C++, communication numérique, traitement de signaux	MATLAB, STM32, hydrophones, PCB

Projet :

SYSTÈME DE MESURE DU COUPLE

Ce projet, préparé par des étudiants, est un bon exemple d'application de la méthodologie en conception et de présentation des outils utilisés.

Le génie pour l'industrie
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

UNIVERSITÉ DU QUÉBEC

RAPPORT FINAL

DANS LE CADRE DU COURS ELE795-HIV20
SYSTÈME DE MESURE DU COUPLE D'Omer 12

PRÉSENTÉ À
LYNE WOODWART, ING., M. Sc. A., Ph. D.

PAR
Charles-Éric Lepage, LEPC19029502
Samuel Deschenes, DESS03069400
Rémy Rodrigue, RODR22079601

MONTRÉAL, 7 avril 2020

Remerciements

En premier lieu, nous aimerions remercier Mme la professeur Lyne Woodward du département de génie électrique pour nous avoir supervisés et conseillés dans notre projet de fin d'études.

Nous tenons par la même occasion à souligner l'aide et les conseils apportés par le professeur et anciens membres de l'équipe d'Omer, M. Christian Belleau du département de génie mécanique.

Nous remercions aussi M. Youssef Bekbouti technicien au département de génie électrique pour son aide dans la fabrication du prototype et ses conseils pour notre banc de test.

Une mention particulière doit être faite à William Mary-Pouliot membre du club Omer pour l'aide fournie dans l'usinage de notre prototype et cela malgré les quelques changements de dernières minutes.

Nous tenons aussi à souligner la participation du club Omer et son représentant Alexandre Plouffe pour leur support financier et la proposition du projet.

Résumé du rapport

Ce rapport se veut la présentation du processus de conception de notre projet de fin d'études en génie électrique. Le projet a consisté au développement d'un capteur de couple pour le Club étudiant Omer de l'ÉTS. Ce capteur est destiné à mesurer le couple que les pilotes transmettent au sous-marin à propulsion humaine en pédalant. Le présent rapport regroupe en détail les informations du processus de développement du capteur développé au cours de la dernière session.

Table des matières

Liste des tableaux

Liste des figures

Liste des équations

1. Introduction

1.1. Mise en contexte

Tel que détaillé dans le rapport de proposition de projet, le club étudiant Omer développe actuellement leur 12^{e} prototype de sous-marin à propulsion humaine. Champion du monde depuis 4 ans, le club vise continuellement à se démarquer en dépassant ses limites. C'est avec cet objectif en tête que les membres du club ont proposé ce projet afin de récolter des données, jusqu'à maintenant estimées lors du développement de leur sous-marin. Ultimement, le club étudiant souhaite optimiser le développement de leur sous-marin ainsi que ses stratégies de course.

1.2. Problématique

Le club étudiant Omer fait actuellement face à un problème de caractérisation de ses sous-marins. Ils n'ont aucun moyen de valider leur modèle théorique durant les essais ou même de confirmer la performance de leurs pilotes en cours de compétitions. C'est pourquoi le club veut équiper ses nouveaux sous-marins d'un système de télémétrie capable de recueillir diverses informations visant à valider sa conception et la performance de leurs pilotes. L'une des données que le regroupement étudiant cherche à obtenir est la mesure du couple généré par ses pilotes.

Notre projet de fin d'études vise donc précisément à développer un système qui permettra à Omer de pouvoir mesurer cette donnée. L'acquisition de cette donnée permettra d'ajouter la force des pilotes aux autres données utilisées pour sélectionner ceux qui prendront place dans le sous-marin en compétition. Il est important de rappeler que la douzième mouture du sous-marin accueillera deux pilotes et qu'il est encore plus important de connaître leur apport individuel au sous-marin, car ceux-ci devront transmettre la même quantité d'énergie aux hélices pour optimiser la transmission de l'énergie totale. De plus, les données du couple réel permettront d'optimiser le design de différentes composantes du sous-marin.

1.3. Objectifs

L'objectif principal de ce projet porte sur le développement d'un concept permettant de mesurer le couple généré par le ou les pilotes du sous-marin sur un arbre rotatif. Le club étudiant pourra, par la suite, reprendre notre travail pour le peaufiner et l'adapter afin de l'intégrer au sous-marin. De cette façon, ils pourront enfin récolter les données du couple généré par leurs pilotes. Nous visons donc le développement d'un prototype conceptuel fonctionnel afin de guider Omer lors du développement et l'intégration du capteur final. Malgré le fait que nous ne développons pas un produit «plug and play», nous devons garder à l'esprit l'intégration de notre concept et les contraintes qu'une utilisation sous-marine dans un habitacle restreint apporte.

1.4. COVID-19

La situation de crise dans laquelle la COVID-19 nous a plongé a eu un impact significatif sur l'étape de mise à l'essai de notre prototype. Le vendredi 13 mars dernier, l'ÉTS a été forcée de fermer les portes de ses établissements selon l'ordre du gouvernement provincial voulant limiter la propagation de la pandémie au Québec.

Au cours de la semaine qui a précédé cette annonce, nous avions mis les efforts nécessaires à la réalisation de notre banc d'essai. Lors de la fermeture de l'école, notre arrangement de tests était prêt, pour qu'au début de la semaine qui suivait, nous puissions mettre à l'essai notre prototype de capteur de couple.

Il est maintenant évident qu'à la lecture de ce rapport, les essais physiques prévus n'ont pu être réalisés étant donné que les bâtiments de l'école sont encore inaccessibles aux étudiants. Étant donné le grand nombre d'équipements spéciaux nécessaires et la limitation des rassemblements sociaux, nous n'avons pas réussi à tester physiquement notre prototype. À noter que lorsqu'on mentionne tests ou essais physiques, nous faisons référence aux essais dans des conditions semblables à l'utilisation prévue du capteur. C'est-à-dire en rotation et soumis à un couple. Ce couple que notre banc d'essai aurait pu simuler de façon stable et connue.

2. Revue de la documentation

Cette deuxième section de rapport présente les concepts théoriques en liens au projet ainsi que les sources documentaires associées.

2.1. Couple

Le concept théorique principal au projet est bien entendu celui du couple. C'est autour de ce concept que le projet en entier est basé. Plusieurs définitions sont disponibles pour définir ce concept. Sa définition générale est qu'on appelle couple l'ensemble des forces appliquées sur un solide dont la résultante est nulle, mais dont le moment total n'est pas nul.[1] Ce moment résultant tend à changer l'état de mouvement radial du solide. La notion de couple peut être résumée comme étant la force qui provoque la rotation d'un objet autour d'un point central tel un arbre[2] d'entraînement de moteur par exemple.

2.2. Torsion

Le concept de torsion fait référence à la « Déformation donnée à un corps allongé par une rotation dans le sens transversal »[3]. La torsion, ou déformation axiale, d'un axe rotatif, tel un arbre d'entraînement, est causée par le couple qu'il subit pour atteindre ou garder son état de mouvement. La torsion que subit un arbre d'entraînement est proportionnelle au couple qui lui est appliqué.

2.2.1. Déformation angulaire[4]

Cette torsion que subit un arbre peut se mesurer selon sa déformation angulaire (α). L'angle de déformation se calcule comme suit :

$$\alpha = {}^{T\,L}/_{G\,J}$$

$$O\grave{u}\ J = \frac{\pi}{32}(De^4 - Di^4)$$

Équation 1- Relation déformation angulaire et couple

et :
T = Couple (Nm)
L = Longueur de l'arbre (m)
G = Module d'élasticité (Pa)
De = Diamètre externe (m)
Di = Diamètre interne (m)

2.3. Résolution / Sensibilité[5]

Les concepts de résolution, sensibilité et précision sont utilisés dans différents domaines technologiques et sont souvent confondus. Ils peuvent cependant signifier la même chose, dans un cas précis.

---

[1] (« Couple (physique) | Wikipédia », 2019)
[2] (Collins English Dictionary, s. d.)
[3] (Larousse, s. d.)
[4] (Note de cours MEC329, Van Ngan Lê & Henri Champliaud, Août 2019)
[5] (« Qualité métrologique d'un appareil de mesure », 2019)

2.3.1. Résolution

La résolution fait référence à la plus petite division d'une quantité unitaire. Pour une plage donnée, plus sa résolution est grande, plus petites sont ses partitions (divisions). Il suffit de penser au nombre de pixels d'un appareil photo. Cette donnée technique spécifie le nombre de pixels (le nombre de points) en combien la photo a été divisée lors de la capture. Plus la résolution est grande plus la photo dans ce cas est découpée en une multitude de points. Dans le cas de la vitesse d'un compteur comptant les coups d'une horloge, battant à X Hz, sur une période de 1 seconde. On dira que la résolution correspondra à celle de l'horloge soit X impulsions.

2.3.2. Sensibilité

La sensibilité quant à elle fait référence au « paramètre exprimant la variation du signal de sortie d'un appareil de mesure en fonction de la variation du signal d'entrée »[6]. En bref, la sensibilité fait référence à la grandeur minimale de variation à l'entrée d'un système pour en voir l'effet à la sortie de celui-ci. Pour reprendre l'exemple de la vitesse à laquelle un compteur peut compter, on dira qu'il est aussi sensible que sa résolution s'il compte à chaque coup d'horloge alors que s'il ne compte qu'à tout les y coups d'horloge sa sensibilité sera de 1/y celle de l'horloge.

[6] (*Accuracy, Precision & Resolution : Electronic Measurements*, s. d.)

3. Méthodologie de travail

La section de la méthodologie sert à décrire les étapes du projet pour répondre aux objectifs. Les étapes sont décrites dans cette section ainsi que listées en détail dans le diagramme de Gantt à l'annexe 1.

3.1. Conception du capteur

Premièrement, nous avons exploré différentes solutions pour mesurer le couple. Nous avons effectué des recherches sur différentes technologies déjà existantes sur le marché. Christian Belleau, professeur au département de génie mécanique, nous a assistés pour les concepts mécaniques autour du couple ainsi que l'aspect électronique. Les solutions retenues répondent aux contraintes physiques définies par le club Omer. Une fois les technologies potentielles ciblées, nous avons sélectionné des composantes pour les commander dès que possible.

Deuxièmement, une fois les technologies choisies, nous avons conçu un banc d'essai pour les tester. Le banc d'essai permet de mesurer le couple à la vitesse nominale. Les pièces mécaniques ont été usinées par les membres du club Omer. Dans le cas des pièces en plastique, elles seront imprimées par l'imprimante 3D de Charles-Éric Lepage et celle des clubs étudiants. Le banc d'essai est aussi composé des modules Lab-Volt situés dans le local A-2504. Youssef Bekbouti du département électrique nous a assistés dans la réalisation du banc de test.

3.2. Réalisation des tests

Une plateforme de développement devait être choisie pour l'aspect logiciel des tests. Nous savions que le club Omer utilisait déjà Arduino, mais qu'ils avaient atteint les limites de cet écosystème. Nous leur avons donc présenté une nouvelle famille de microprocesseur. Christian Belleau nous a guidés dans ce choix. Une période de familiarisation avec le nouveau logiciel de programmation avait été prévue.

Ensuite, avec un banc de test assemblé et programmé, nous devions procéder aux essais. Nous devions effectuer un processus itératif dans le but d'éliminer des solutions non fonctionnelles ou optimiser une des solutions prometteuses. Si le temps le permettait, nous devions fournir à notre client un circuit imprimé et une programmation conçue pour être intégrée au système embarqué du sous-marin. L'objectif principal du projet restait tout de même d'identifier la meilleure solution pour mesurer le couple.

4. Processus de conception

Comme expliqué dans la revue de la documentation, lorsqu'un couple est appliqué à un arbre de transmission, cet arbre se déforme de façon proportionnelle au couple. Cette déformation mécanique peut-être mesurée via différentes technologies. C'est ainsi que nous avons développé différents concepts utilisant des technologies différentes basées sur les contraintes du projet.

4.1. Les contraintes du projet

À la suite du rapport préliminaire, nous avons identifié les contraintes suivantes pour la réalisation du capteur de couple :
Tout d'abord le système devra pouvoir supporter un couple maximal de 70 Nm et il devra pouvoir fournir une lecture précise au dixième de Newton mètre à 100 RPM avec un couple nominal de 3.5 Nm. L'arbre de transmission pour l'élaboration du capteur aura un diamètre de 20mm.

Les concepts envisagés ne devront pas être trop gros. Le tout devrait pouvoir tenir dans une boite de 2 po d'épaisseur, 8 po de long et 4 po de large, mais idéalement les dimensions devront être plus petites que ça. De plus, la conception du capteur devrait permettre de facilement rendre le tout étanche dans une prochaine itération. Finalement, le coût de production d'un capteur ne devrait pas dépasser 150$.

5. Concepts étudiés

5.1. La jauge de déformation

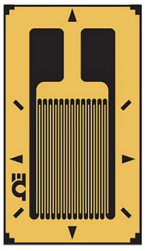


Figure 1- Modèle d'une gauge de déformation

Ce premier concept se base sur la technologie des jauges de déformation. En plaçant une jauge à 45° par rapport à l'axe de rotation, la jauge va se déformer en suivant l'effet de torsion que subit l'arbre sous un couple et cela de façon proportionnelle.

Puisque les jauges de déformation sont en fait des résistances qui varie proportionnellement à la déformation qui leur est appliqué, il est donc possible de mesurer un courant qui varie selon la déformation.

Cependant, la lecture d'un courant est plutôt fastidieuse particulièrement dans le cas d'une jauge ou les variations de résistance sont très faibles. C'est pourquoi nous utiliserions un montage en pont de Wheatstone qui permettrait de mesurer la variation de tension pour une petite variation de résistance. Cette tension serait alors amplifiée à l'aide d'un amplificateur d'instrumentation et mesurée par un microcontrôleur. En testant chaque système sous des conditions connues, il serait alors possible d'obtenir le couple en fonction de la tension.

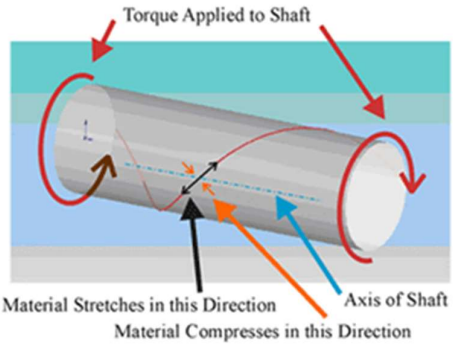


Figure 2 - Contrainte sur un arbre de transmission

Pour ce qui est des contraintes, il suffit de sélectionner une jauge qui respecte nos critères de couple. En revanche au niveau des dimensionnements, le fait d'avoir un émetteur et un récepteur rend la tâche plus complexe. Particulièrement avec l'émetteur qui tourne autour de l'arbre. Financièrement, le projet est réalisable malgré le coût d'achat initial des jauges puisque le coût du reste des composantes analogiques et numériques est faible.

L'utilisation de ce concept apporterait au club Omer une solution fiable qui a mainte fois été testée et éprouvée dans de multiples domaines pour la mesure du couple. Par contre, bien que cette technologie ait déjà prouvée son fonctionnement, l'intégration d'une jauge de déformation sur un arbre tournant implique la fabrication d'un système de communication sans-fil et de concevoir un système d'acquisition analogique. La réalisation d'un émetteur et d'un récepteur sans-fil est complexe et la fabrication d'un PCB qui n'interfère pas avec les signaux analogiques est tout un défi. C'est pour ses raisons que les risques que le projet ne fonctionne pas dans le temps alloué sont relativement élevés.

De plus, l'installation d'une jauge de contrainte sur un arbre est relativement complexe et permanente. Il est donc impossible de remplacer une jauge mal installée ou endommagée. Finalement, puisqu'une jauge à un coût d'achat important d'environs 250$, le club Omer ne peut prendre le risque financier d'en endommager et de devoir en acheter plusieurs.

5.2. L'encodeur optique

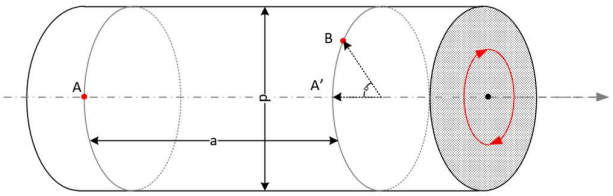


Figure 3 - Positionnement et dimensionnement

L'idée avec ce concept c'est de mesurer physiquement l'angle de déformation ALPHA sur une distance déterminé (a) pour un arbre de diamètre (d) connu. Bien que cette idée semble complexe à priori, il suffit de connaître la position angulaire aux points A et B.

Il existe déjà plusieurs techniques pour obtenir ces informations, la plus commune est probablement l'encodeur optique. Cet appareil fixé à l'arbre de transmission envoie une impulsion à chaque déplacement de X degrés. Où X est la définition de l'encodeur. Ainsi en comptant le nombre d'impulsions, il est facile de déterminer la position de l'encodeur en degré.

La méthode utilisée ici consisterait à installer deux encodeurs séparés par une distance (a) connue. En faisant la différence de la position des deux encodeurs, nous obtiendrions donc la déformation angulaire ALPHA sur la distance (a). Chaque encodeur serait connecté à un microcontrôleur ayant la capacité de mémoriser et de compter les impulsions sans perte.

Cependant la précision requise pour mesurer alpha qui est très petite demanderait des encodeurs dispendieux. De plus, les dimensions des encodeurs pour un arbre de transmission de 20mm dépassent les dimensions maximales demandées.

Pour ce qui est des risques, ce type de système a déjà fait ses preuves. Les encodeurs optiques sont conçus pour être utilisés dans des conditions difficiles durant de nombreuses heures. De plus, leur simplicité de fonctionnement et leur polyvalence les rend facilement intégrables à tout type de domaine, le risque technologique est quasiment nul.

5.3. Le compteur de temps

Le concept ici diffère un peu, mais découle du même principe que l'encodeur. Mais au lieu de constamment connaître la position de deux points sur l'arbre, nous calculerons le temps qui s'écoule entre le passage du point A et du point B (Voir Figure 3).
Pour mesurer ce temps, deux capteurs de position sur le même horizon et séparés par une distance (a) généreraient une impulsion à chaque passage du point devant celui-ci. Le front montant du point A déclencherait un minuteur d'un microcontrôleur et le front montant du point B arrêterait le minuteur. Avec le temps entre chaque front et la vitesse de rotation obtenue via un second minuteur déclenché et arrêté par le point A, il est possible de connaître la distance entre le point A et B qui les sépare sur l'arc de cercle de l'arbre de transmission. Ainsi, avec un peu de géométrie de base, il est possible de déterminer l'angle de déformation ALPHA et déduire le couple qui lui est associé.

Ce concept aurait l'avantage d'être peu dispendieux. La plus grosse dépense serait probablement la fabrication du circuit imprimé (20$) et l'achat de capteur ayant la capacité de fonctionner à une fréquence nominale d'environs 2 Hz (120 RPM / 60 sec = 2 Hz). Pour ce qui est du couple maximal puisqu'aucun contact mécanique n'est présent dans ce concept, le couple pourrait être infini.

Bien entendu ce concept est plus risqué. C'est une technique qui n'est pas couramment employée et qui se base sur un microcontrôleur rapide, autour de 80 kHz. Les risques que le projet ne soit pas fonctionnel dans les temps alloués sont plus grands. Cependant, les risques financiers sont moindres et le concept utiliserait des pièces peu coûteuses et faciles d'accès.

5.4. Choix du concept retenu

Pour faire le choix de la meilleure technologie pour mesurer le couple, nous avons utilisé la matrice de décision ci-dessous. Chaque critère est noté sur dix puis pondéré avec un pourcentage selon son importance.

Tableau 1 - Matrice de décision

		Solution 1		Solution 2		Solution 3	
		Jauge de contrainte		Encodeur Optique		Compteur de Temps	
	Poids %	Score /10	%	Score /10	%	Score /10	%
Critères physiques	**25%**						
Dimensions	5,0%	9	4,5%	7	3,5%	6	3,0%
Maintenance	5,0%	6	3,0%	8	4,0%	8	4,0%
Facilité d'assemblage	10,0%	5	5,0%	7	7,0%	6	6,0%
Fiabilité mécanique	5,0%	3	1,5%	8	4,0%	9	4,5%
Critères économiques	**14%**						
Coûts des pièces	12,0%	5	6,0%	4	4,8%	9	10,8%
Coûts d'implémentation (post projet)	2,0%	6	1,2%	6	1,2%	8	1,6%
Critères temporels	**33,5%**						
Temps de fabrication	8,0%	4	3,2%	7	5,6%	6	4,8%
Temps de programmation	8,0%	9	7,2%	7	5,6%	5	4,0%
Complexité de conception	10,0%	7	7,0%	7	7,0%	5	5,0%
Disponibilité des pièces	7,5%	10	7,5%	8	6,0%	10	7,5%
Critères de fonctionnalités	**27,5%**						
Précision	15%		0,0%		0,0%		0,0%
Facilité d'interfaçage	8,5%	2	1,7%	9	7,7%	7	6,0%
Compétence de club (post projet)	4,0%	6	2,4%	7	2,8%	6	2,4%
Totaux:	**100,0%**		**47,8%**		**56,4%**		**57,2%**

La matrice n'a pu être complétée, car les essais qui devaient valider les critères de fonctionnement n'ont pu être réalisés. Cependant elle peut toujours guider vers les solutions qui sont davantage prometteuses. À sa lecture nous pouvons, en effet, exclure la jauge de contrainte à cause de son coût élevé et de sa complexité d'interface. Il nous reste donc la solution de l'encodeur. Nous devions tester deux types de capteurs différents. Le premier était un capteur optique et le second était à effet Hall.

Nous devions par la suite utiliser le principe du compteur de temps décrit dans la section précédente. Le capteur à effet Hall et l'encodeur optique produisent tous les deux des fronts montants nécessaires. Il nous aurait suffi de déterminer la précision que nous pouvons aller chercher entre les deux fronts. Malheureusement, nous n'avons pas réussi à effectuer les essais nécessaires afin de déterminer lequel de ces deux capteurs aurait été le plus précis. Cependant, nous supposions que le capteur optique aurait donné de meilleurs résultats étant donné la plus petite latence associée à ce genre de capteur en général. Cette supposition n'aura pas pu être

testée. Nous croyons toutes fois que le concept d'encodeur, peu importe la technologie utilisée, reste la meilleure option tel que le suggère la matrice de décision.

5.5. Outils et techniques d'ingénierie utilisés

Pour réaliser ce capteur, il nous a fallu utiliser différents outils d'ingénierie, dont des logiciels et plateformes de développement. Ci-dessous sont listés ces outils avec le numéro de modèle/version utilisé ainsi qu'une brève description de leur utilité.

5.5.1. Altium Designer 2020

Altium Designer est un logiciel de conception de circuit imprimé de niveau professionnel. Grâce à ce logiciel, nous avons conçu les schématiques du circuit électrique et développer un circuit imprimé. L'avantage de ce logiciel, c'est qu'il permet de générer avec peu de difficulté un fichier STEP qui contient le modèle 3D du circuit imprimé qui peut être importé vers n'importe quel logiciel de CAO moderne.

5.5.2. Inventor 2020

Inventor est un logiciel de modélisation 3D développé par Autocad. Avec ce logiciel, nous avons développé la structure du prototype, ainsi que l'arbre de transmission pour le banc de test. Il nous a permis aussi de concevoir l'assemblage complet de notre prototype et de valider les dimensions et le positionnement du circuit imprimé par rapport à l'arbre de transmission.

5.5.3. STM32CubeMX

STM32CubeMX est un logiciel de programmation utilisé pour des microprocesseurs 32 bits produits par la compagnie STMicroelectronics. Ce logiciel est basé sur l'ancienne plateforme Eclipse qui permet de programmer en C pour des systèmes avec un processeur ARM Cortex. Il contient une interface graphique pour faciliter la configuration des périphériques. Nous avons utilisé la librairie HAL qui supporte toutes les familles STM32.

5.6. Outils et techniques créés pour ce projet

Pour réaliser ce capteur, il nous a fallu créer différents outils d'ingénierie, dont des calculateurs .xlsx, un banc d'essai ainsi qu'un programme pour notre microcontrôleur. Ci-dessous sont listés ces outils ainsi qu'une description de leur utilité.

5.6.1. Bancs de tests

Afin de vérifier notre concept de capteur ainsi que ses performances, il nous a fallu monter de toute pièce, un banc de test. Celui-ci devait nous permettre de confirmer les concepts techniques utilisés ainsi que le bon fonctionnement de notre capteur entier, soit les sections physiques, l'électronique et la programmation. Le banc de tests est présenté plus en détail au cours de la section Banc d'essais du présent rapport.

5.6.2. Calcul de la résolution de la minuterie du compteur de temps

Pour avoir une résolution acceptable, la fréquence à laquelle la minuterie doit compter. Un critère de résolution n'a pas été défini dans les contraintes de conception. Nous avons développé un chiffrier Excel présenté à l'annexe 2 pour calculer cette précision. Pour avoir une résolution supérieure 1/10 pour la plage de 0-3.5Nm, la fréquence du microcontrôleur doit être supérieure à 5 MHz et 32 bits. Ce chiffrier nous permet de trouver la relation entre la résolution et la vitesse de rotation et la fréquence de la minuterie. Les Figure 4 et Figure 5 présentent ces relations.

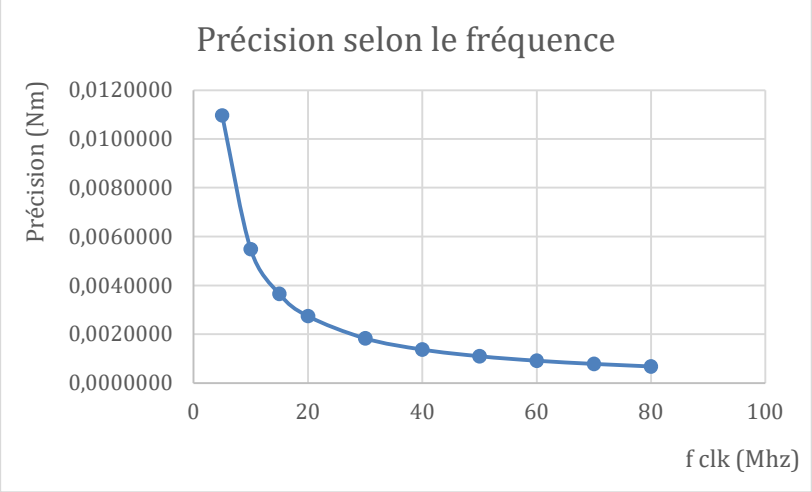


Figure 4 - Précision du capteur selon la fréquence de la minuterie

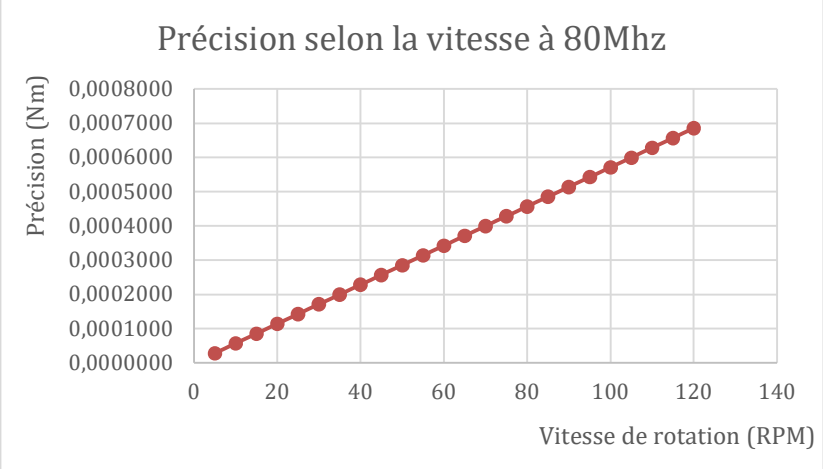


Figure 5 - Précision du capteur selon la vitesse de rotation

Nous avons déjà un microcontrôleur qui correspond à ce critère soit la plateforme physique NUCLEO-L476LG avec une horloge de 80Mhz. Cette fréquence nous permet d'avoir plus 6128 incréments pour la plage nominal du couple à 100 RPM.

6. Résultats obtenus

6.1. Présentation du prototype développé

Basés sur nos choix de capteur, nous avons développé un prototype fonctionnel conçu pour être installé à proximité d'un arbre de transmission, mais sans contact avec ce dernier, d'un design relativement simple qui implique peu de modifications sur l'arbre de transmission. Le prototype est donc en deux parties, un circuit imprimé et l'arbre de transmission.

6.1.1. Le circuit imprimé

Le design du circuit électronique est relativement simple. Grossièrement, il est composé de deux capteurs optiques A et B qui peuvent être alimentés entre 3.3V et 24V. Chaque capteur retourne un signal binaire, zéro volt ou la tension d'alimentation de 5V. Un niveau logique bas signifie qu'il y a une présence dans le capteur optique et un niveau logique haut signifie l'état inverse. Vous retrouverez à l'annexe 3 les schématiques du circuit imprimé.

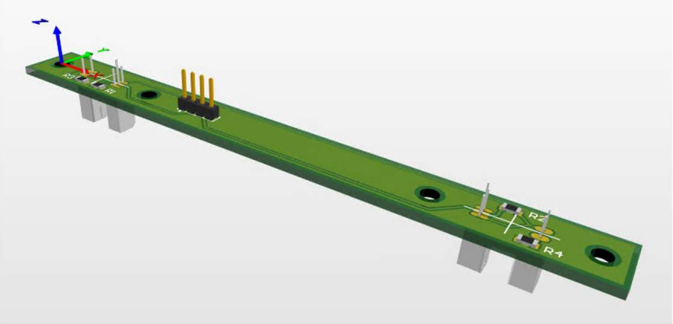


Figure 6 - Rendu numérique du circuit imprimé

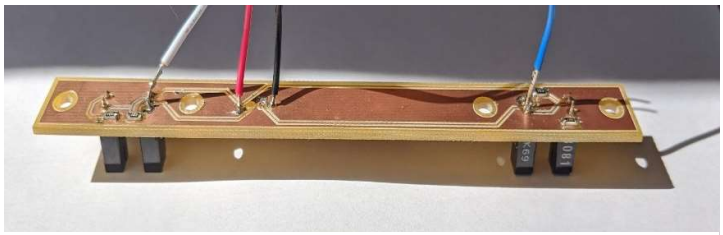


Figure 7 - Résultat final du circuit imprimé

Le défi du circuit imprimé résidait dans sa conception mécanique. Il était impératif de concevoir le circuit imprimé dans l'idée qu'il soit le plus minimaliste et que le positionnement des capteurs soit bien défini pour nous permettre de placer le capteur le plus proche possible de l'arbre de transmission sans avoir de contact et de connaître la distance exacte entre les deux capteurs (la distance "a" décrite dans la section parlant de l'encodeur optique). Nous en sommes arrivés à un concept qui fait 15 mm x 150 mm. La distance "a" a été fixée à 10 cm (Figure 3)

6.1.2. L'arbre de transmission

Avec le choix du capteur et la conception du circuit imprimé, cela implique la fabrication de 2 trous distancés de 10 cm (la distance "a", Figure 3) sur l'arbre de transmission. Ces trous sont utilisés pour incruster des "Spring Pin". Ce sont ses pins qui passent devant le capteur pour générer une variation dans leur état logique.

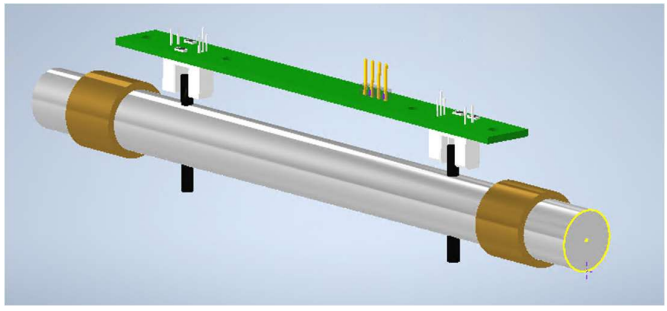


Figure 8 - Rendu de l'assemblage du capteur

Pour finir, ce prototype permet de facilement tester notre concept en le raccordant à un microcontrôleur approprié. Via un connecteur de type "header" à 4 pins.

6.1.3. Structure du programme

Le programme est implémenté par le logiciel STM32CubeMX sur la plateforme de développement NUCLEO-L476G. Le concept qui est exploité pour trouver le couple de l'arbre de transmission est le compteur de temps. Pour réaliser ce concept, deux capteurs de positions sont placés à chaque bout de l'arbre. Un capteur sert de référence et le temps est compté pour trouver l'angle de déformation. Le lien entre l'angle de déformation et le couple de l'arbre est décrit par l'Équation 1-

Le programme a donc deux intrants soient les deux capteurs ainsi qu'un extrant qui est l'ordinateur pour recueillir les données. Le programme devait accomplir plusieurs fonctionnalités décrites ci-dessous. Le reste le cette section du rapport explorera comment le programme est structuré pour réaliser ces fonctionnalités.

Fonctionnalités du programme :

- Capturer le moment où la première dent passe devant le capteur;
- Calculer la vitesse de rotation;
- Capturer le moment ou la dent opposée sur l'arbre de transmission passe au deuxième capteur;
- Calculer l'angle de déformation puis le couple;
- Transmettre le résultat sur un port série.

Le programme comprend les fichiers main.c, t_torque.h et t_torque.c qui se trouvent à aux Annexes 4, 5 et 6. Le fichier main gère la configuration des périphériques, les interruptions et le programme principal. Le fichier t_torque sert aux calculs du couple de l'arbre.

Le comptage du temps des deux capteurs est effectué par le même minuteur du microprocesseur. Il existe deux minuteurs avec la résolution de 32 bits désirée: les minuteurs 2 et 5. Le minuteur 5 est déjà utilisé par le périphérique de transmission série alors, seulement la minuterie 2 est utilisée. À chaque passage d'une dent au premier capteur, la valeur du compteur de la minuterie est capturée, puis le compteur est mis à zéro. Cette première capture sert à calculer la vitesse de rotation de l'arbre. Au passage de la dent opposé du deuxième capteur, le compteur est capturé pour calculer l'angle de déformation et ultimement le couple. La gestion de la minuterie est faite par interruption pour éviter que le temps de calcul affecte les capteurs. La Figure 9 ci-dessous illustre le fonctionnement général du capteur.

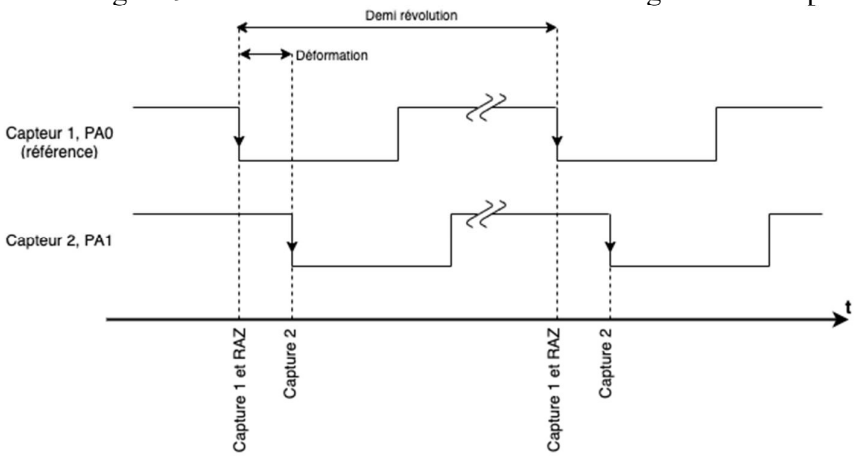


Figure 9 - Chronogramme du capteur

Pour réaliser ces fonctionnalités, la minuterie 2 devra être configurée en mode de capteur d'entrées. Dans ce mode, la minuterie va incrémenter à la fréquence de l'horloge principale à 80 MHz. Les capteurs 1 et 2 sont respectivement connectés aux entrées PA0 et PA1 du microprocesseur. Ces entrées correspondent aux canaux de capture 1 et 2 de la minuterie. Des filtres peuvent être configurés sur chaque entrée. Ils correspondent au nombre de coups d'horloge pendant lesquels l'entrée doit être à un niveau bas avant de lever un drapeau pour l'interruption. Ces filtres devront être ajustés lors des tests complets. Les figures X et X montrent comment les paramètres sont configurés dans l'interface graphique de STM32CubeMX. La configuration des registres du périphérique de la minuterie 2 est automatiquement générée par le logiciel.

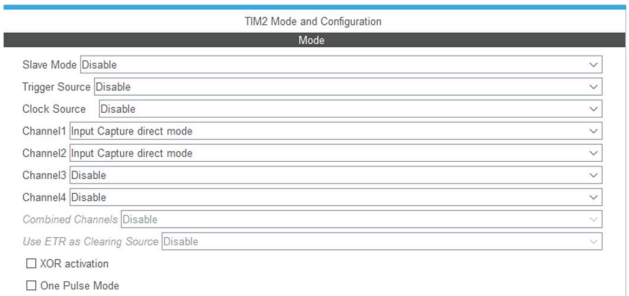


Figure 10 - Mode et configuration TIM2

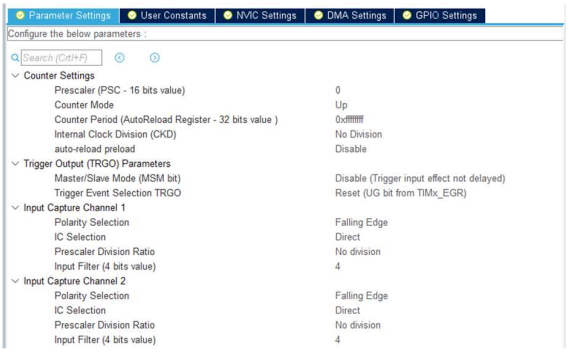


Figure 11 - Configuration compteur

Les détails se retrouvent dans le programme main.c à l'annexe 4dans les fonctions `MX_TIM2_Init` pour la configuration et `HAL_TIM_IC_CaptureCallback` pour l'interruption.

Nous allons voir comment une donnée sera représentée dans le programme. Une donnée est mise dans une structure nommée *t_torque* (Figure 12) et contenant plusieurs membres. La structure se retrouve dans le fichier t_torque.h. Le *data_id* sert à numéroter les données. La numérotation commence à zéro puis incrémente de 1 à chaque nouvelle donnée. Les membres *raw_speed* et *raw_angle* contiennent les valeurs brutes des captures de la minuterie. Le *speed_degree*, *angle* et *torque* sont les résultats de leur calcul respectif. Le membre *state* sert à la machine d'états décrite plus loin.

```
    // Données d'une prise de mesure du torque
    typedef struct{

        uint32_t data_id;       // Numérotation séquentielle de la mesure du couple
        uint32_t raw_speed;     // Valeur brut du timer pour la vitesse
        uint32_t raw_angle;     // Valeur brut du timer pour l'angle de déformation
        float speed_degree;     // Vitesse de rotation de l'arbre en degrée par sec
        float angle;            // Angle de déformation de l'arbre
        float torque;           // Torque en Nm
        int8_t state;           // Status de la donnée

}t_torque;
```

Figure 12 - Structure t_torque

Le programme est structuré par une machine d'état représentée à la Figure 13. La machine d'état se retrouve dans la boucle while(1) dans le fichier main.c.

Aux états 0 et 2, le programme attend les fronts descendants générés par les capteurs. Les drapeaux qui indiquent ces moments sont levés par l'interruption de la minuterie 2. Les valeurs du compteur sont mises dans la donnée *raw_speed* pour le premier capteur et *raw_angle* pour le deuxième.

Les états 1 et 3 servent pour calculer la vitesse de rotation, l'angle de déformation et le couple. Les Équation 2ci-dessous montrent comment calculer la vitesse de rotation et l'angle de déformation à partir des valeurs du compteur. Le couple se calcule ensuite par l'Équation 1.

$$\omega = (360°/N) \cdot (raw_{speed}/f_{clk})$$

Équation 2 - Vitesse de rotation

$$\alpha = (raw_{angle}/f_{clk}) \cdot \omega$$

Équation 3 - Angle de déformation

$$\omega = vitesse\ de\ rotation\ degrée/s$$
$$N = Nombre\ de\ dents\ autour\ de\ l'arbre$$
$$f_{clk} = fréquence\ de\ l'horloge$$
$$\alpha = angle\ de\ déformation$$

À l'état 4, la donnée est transmise sur le port série. À noter que le périphérique UART est configuré automatiquement par l'interface graphique. L'état 5 permet de remettre la variable de la donnée à zéro pour recommencer à l'état 0.

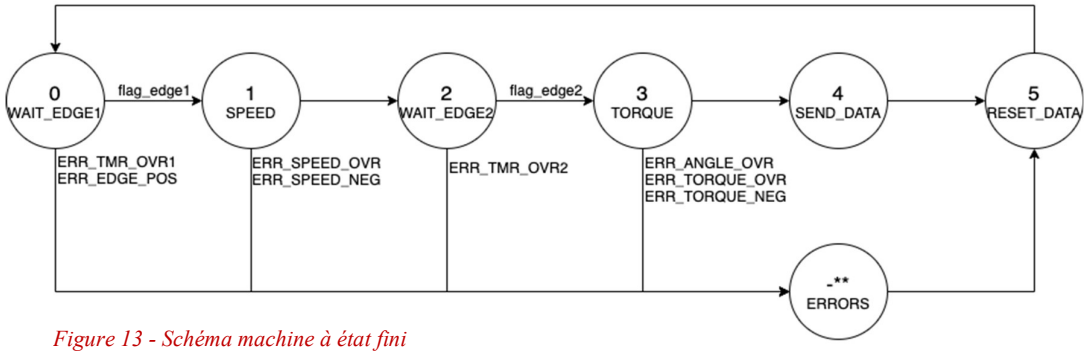


Figure 13 - Schéma machine à état fini

Différents codes d'erreur peuvent être générés par le programme. Toutes les erreurs ont un code négatif et la dizaine correspond à l'état où elles sont générées. L'erreur est par la suite transmise sur le port série. Le ci-dessous indique tous les codes d'erreur possible.

Tableau 2- Code d'erreur

Code	Nom de l'erreur	Description	Seuil
-1	ERR_TMR_OVR1	Dépassement de la valeur maximale de la minuterie en attendant le 1er capteur.	htim2.Init.Period = 0xffffffff
-2	ERR_EDGE_POS	Le 2e capteur est arrivé avant le 1er.	-
-11	ERR_SPEED_OVR	La vitesse est physiquement impossible.	ERR_SPEED_OVR_VALUE
-12	ERR_SPEED_NEG	L'arbre tourne dans le sens inverse.	0
-21	ERR_TMR_OVR2	Dépassement de la valeur maximale de la minuterie en attendant le 2e capteur.	htim2.Init.Period = 0xffffffff
-31	ERR_ANGLE_OVR	L'angle de déformation est physiquement impossible.	ERR_ANGLE_OVR_VALUE
-33	ERR_TORQUE_OVR	Le torque est trop grand.	ERR_TORQUE_OVR_VALUE
-34	ERR_TORQUE_NEG	Le pilote force contre le sens qui le fait avancer.	0

6.2. COVID-19

Malgré les limitations apportées par le confinement social apporté par le coronavirus, nous avons tout de même effectué certains essais. Nous avons principalement validé la section logiciel (programmée) de notre capteur. La section suivante présentera notre banc d'essai et les détails de la méthode d'essais logiciels ainsi que les résultats seront présentés aux sections subséquentes.

6.3. Banc d'essai

Cette section a été rédigée à l'imparfait afin de mettre de l'avant le fait que nous n'avons pas réussi à nous servir du banc d'essais.

Notre banc d'essai devait pouvoir appliquer un couple connu à un arbre tournant à environ 100 rotations par minutes (RPM). Pour atteindre ces objectifs, nous avons décidé d'utiliser 3 modules dynamométriques LabVolt disponibles dans le local de laboratoire A-2504. Un module a été utilisé comme moteur pour faire tourner l'arbre. La vitesse de rotation était ajustée selon la tension d'alimentation CC qui lui était appliquée. La vitesse de rotation était donnée sur l'afficheur du module. Les deux autres modules étaient utilisés comme charges mécaniques ajustables. Celles-ci étaient connectées en parallèle à l'arbre rotatif via une courroie. On pouvait ajuster l'intensité de la charge mécanique, c'est-à-dire le couple qu'il appliquait via un potentiomètre. Le couple appliqué pouvait être lu sur l'afficheur de chacun des deux modules. L'alignement des modules entre eux était assuré par un arrangement physique de contreplaqué dans lequel les modules s'inséraient. Nous n'avons malheureusement pas de photos, car nous attendions de confirmer le fonctionnement du banc d'essai avant de le documenter plus en détail dans ce rapport. Nous pouvons tout de même insérer, ci-dessous, le schéma utilisé dans le rapport d'étape pour illustrer le concept.

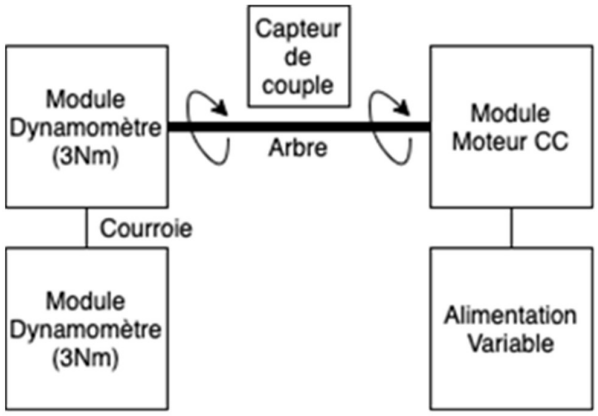


Figure 14 - Schéma bloc du banc de test

6.4. Présentation des résultats

Comme le banc d'essai ne peut être assemblé, le programme a été testé à l'aide d'un circuit qui simule les signaux des capteurs avec l'arbre tournant. Le circuit est composé de deux circuits intégrés 555. L'un est configuré en mode bistable stable produisant une onde périodique simulant la vitesse de l'arbre. L'autre placé à la sortie du premier est en mode monostable produisant un front descendant synchronisé, mais décalé pour simuler l'angle de déformation. Des potentiomètres permettent d'ajuster la vitesse et l'angle. Le circuit décrit se trouve à la figure ci-dessous.

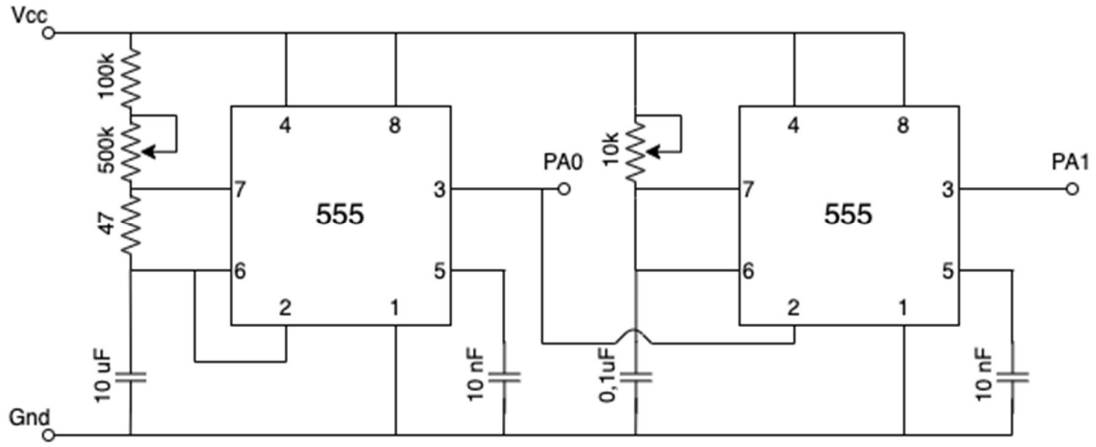


Figure 15 - Circuit de test

À l'aide d'un terminal série, nous avons obtenu les résultats suivants présentés au tableau ci-dessous.

Tableau 3- Résultat du programme

Résultats du programme						
data_id	speed_raw	angle_raw	speed_degree	angle	torque	state
			deg/s	deg	Nm	
0	132137512	45444	297,3	0,000349	1,550	4
1	286878608	45410	645,5	0,000484	2,150	4
2	286874634	45416	645,5	0,000533	2,360	4
3	286961250	45418	645,7	0,000549	2,430	4
4	286550294	45458	644,7	0,00087	3,860	4
5	286844142	45430	645,4	0,000645	2,860	4
6	286805386	45458	645,3	0,000871	3,860	4
7	286871164	45420	645,5	0,000565	2,500	4
8	286934384	45479	645,6	0,001041	4,610	4
9	286880547	45480	645,5	0,001049	4,650	4
10	287004126	45654	645,8	0,002454	10,880	4

6.5. Analyse des résultats de vérification

Avec un chiffrier Excel (Voir tableau ci-dessous), nous pouvons vérifier les calculs du programme. Nous pouvons constater que les erreurs de calcul sont bien en dessous de 1% ce qui est acceptable.

Tableau 4- Validation des résultats

Vérification							
Temps vitesse	Vitesse		Temps angle	Angle	Torque	Erreur vitesse	Erreur torque
s	deg/s	RPM	s	deg	Nm	%	%
1,6517	297,3	49,6	0,000001175	0,000349	1,549	0,00%	-0,09%
3,5860	645,5	107,6	0,00000075	0,000484	2,146	0,00%	-0,18%
3,5859	645,5	107,6	0,000000825	0,000533	2,361	0,00%	0,03%
3,5870	645,7	107,6	0,00000085	0,000549	2,433	-0,01%	0,12%
3,5819	644,7	107,5	0,00000135	0,000870	3,858	0,01%	-0,04%
3,5856	645,4	107,6	0,000001	0,000645	2,861	0,00%	0,04%
3,5851	645,3	107,6	0,00000135	0,000871	3,862	0,00%	0,05%
3,5859	645,5	107,6	0,000000875	0,000565	2,504	-0,01%	0,15%
3,5867	645,6	107,6	1,6125E-06	0,001041	4,615	0,00%	0,11%
3,5860	645,5	107,6	0,000001625	0,001049	4,650	0,00%	0,00%
3,5876	645,8	107,6	0,0000038	0,002454	10,878	-0,01%	-0,02%

Malheureusement, ce test ne permet pas de vérifier le temps compté par la minuterie deux. Nous n'avons pas eu accès un oscilloscope pour vérifier si les valeurs brutes capturées sont valables.

6.6. Retours dur les contraintes du projet

Au tout début de ce projet, nous avions identifié les contraintes qui baliseraient notre projet de développement d'un capteur de couple. Ici, il faut tenir compte de l'impact qu'a eu la COVID-19 sur l'étape de test physique de notre prototype.

Nous n'avons pu tester la précision demandée de 1/10 Nm selon une vitesse de 100RPM, tout comme la lecture du couple à 70Nm. Cependant, comme notre système n'est pas lui-même déformé par le couple mesuré, on être confiant que le couple maximal aurait pu être lu et n'aurait pas causé de bris à notre capteur. La plasticité du l'arbre rotatif est en théorie la limite de couple imposé à notre capteur.

Pour ce qui est de la contrainte de coût de production qui était d'un maximum de 150$, nous avons réussi à la respecter. En effet, bien que notre PCB ait été fabriqué à l'ÉTS, nous pouvons estimer son coût auquel on doit ajouter les composantes électroniques, l'impression 3D, la quincaillerie mécanique et la plateforme de développement du microcontrôleur. Ne pas oublier que ce microcontrôleur peut être utilisé à différentes fins en même temps dans le sous-marin. Nous estimons que le coût total d'un capteur complet serait inférieur à 100$, soit autours de 70$ à 90$. Le tableau 4 ci-dessous présentent les coûts projetés pour l'assemblage d'un ensemble capteur.

Tableau 5- Estimation du coût de production du capteur

Pièces	Coût
PCB	20$ - 25$
Composantes électroniques	10$
Composantes mécaniques	10$
Impression 3D	5$
Plateforme NUCLEO-L476LG	30$ - 40$
TOTAL	75$ - 90$

Les contraintes de dimensionnement quant à elles n'ont pas été respectées pour différentes raisons dont, entre autres, l'indisponibilité d'obtenir un arbre rotatif de 20mm de diamètre. Celles-ci devaient, au départ, être idéalement plus petite que 2 po d'épaisseur, 8 po de long et 4 po de large. Suite à cela, nous avons convenu avec l'équipe d'Omer de nous concentrer sur une preuve de concept qu'il pourra redimensionner à sa guise. Ce qui a eu pour effet de ne pas tenir compte des dimensions imposées. Le concept étant très minimaliste, notre client pourra sans problème le redimensionner.

Conclusion

Notre projet de fin d'études visait le développement d'un capteur de couple pour le club étudiant Omer de l'ÉTS. L'intérêt pour le club est d'obtenir les informations de couple généré par les pilotes dans le but de suivre leur performance et d'optimiser la conception des prochains sous-marins. Ils visent toujours à se dépasser et à innover. C'est exactement pour cette raison qu'on nous a chargés de développement d'un système de mesurage du couple.

Tout au long cette session, nous avons travaillé sur ce projet avec comme objectif de fournir une preuve de concept dont Omer pourra s'inspirer pour éventuellement l'implanter dans leur sous-marin. Le présent rapport a présenté le processus de conception de notre concept de capteur de couple inspiré des encodeurs.

Nous avons réussi à mettre à terme prototype prêt pour des essais. Un circuit imprimé avec des capteurs ainsi qu'un arbre de transmission ont été réalisés. Un programme sur une plateforme de développement STM32 a été testé grâce à un circuit qui simule les signaux des capteurs. Finalement, un banc d'essai a été partiellement assemblé avec des modules LabVolt. Alors, il manquait seulement à combiner ces trois parties pour confirmer la faisabilité du concept du compteur de temps avec des capteurs optiques.

Malgré l'impact important qu'a eu la crise du COVID-19 sur la partie d'essais de notre prototype, nous considérons que nous avons majoritairement atteint notre objectif initial. Nous avons effectué différentes recherches afin de confirmer ou d'infirmer différents concepts et différentes technologies. Plusieurs articles vantaient les mérites du mesurage du couple avec des encodeurs. Nous avons donc choisi de baser notre concept sur celui des encodeurs afin de mesurer une déformation angulaire en torsion de l'arbre d'entraînement soumis à un couple. Il aurait été plus qu'intéressant de tester notre prototype dans des conditions expérimentales, mais nous devons laisser cette étape au membre du club étudiant. Il sera intéressant de voir si notre concept et prototype est fonctionnel et fiable. Les résultats qu'ils pourront en tirer devraient être fort intéressants.

7. Recommandation et poursuite du projet

Tout d'abord, nous recommandons au club étudiant OMER de commencer par tester notre prototype comme nous l'aurions fait, c'est-à-dire avec le banc de test que nous avons préparé. De là, ils pourront déterminer si des modifications sont nécessaires afin d'assurer le bon fonctionnement de notre prototype dans les conditions de test expérimental.

7.1. Implémentation du concept

Suite à la confirmation du bon fonctionnement de notre concept, il faudra que le club OMER adapte le prototype qui a servi à prouver notre concept. Il faudra que les dimensions physiques soient modifiées pour pouvoir s'intégrer au sous-marin. Pour le moment, nous avons utilisé un arbre en aluminium contrairement à l'acier inoxydable. Son diamètre est plus petit et l'empattement de notre capteur est trop long pour pouvoir être monté à l'endroit prévu dans le sous-marin. De plus, il faudra prévoir un boîtier étanche pour l'électronique, sans compter le machinage des arbres pour y intégrer les composantes pour la détection. Il reste donc pas mal de modifications à faire afin de pouvoir tester le capteur dans des conditions d'utilisation réelle.

Nous espérons que le club ira de l'avant avec notre concept et qu'ils réussiront à obtenir les données dont ils rêvent depuis des années.

Bibliographie

1. Couple (physique) | Wikipédia. (2019). In Wikipédia. https://fr.wikipedia.org/w/index.php?title=Couple_(physique)&oldid=160562413

2. Torque definition and meaning | Collins English Dictionary. (s. d.). Consulté 7 février 2020, à l'adresse https://www.collinsdictionary.com/dictionary/english/torque

3. *Définitions : Torsion | Dictionnaire de français Larousse*. (s. d.). Consulté 7 février 2020, à l'adresse https://www.larousse.fr/dictionnaires/francais/torsion/78555

4. Goulet, J.-A. (s. d.). *Module #4 Torsion (CIV1150—Résistance des matériaux)* [Cours CIV1150 - Polytechnique Montréal].

5. Accuracy, Precision & Resolution : Electronic Measurements. (s. d.). Consulté 7 février 2020, à l'adresse https://meettechniek.info/measurement/accuracy.html

6. Qualité métrologique d'un appareil de mesure. (2019). In Wikipédia. https://fr.wikipedia.org/w/index.php?title=Qualit%C3%A9_m%C3%A9trologique_d%27un_appareil_de_mesure&oldid=164947128

7. Jauge de déformation. (2019). In Wikipédia. https://fr.wikipedia.org/w/index.php?title=Jauge_de_d%C3%A9formation&oldid=161117934

8. SGD-LINEAR1-AXIS | Precision Strain Gages—1-Axis General Purpose | Omega. (s. d.). Consulté 10 avril 2020, à l'adresse https://www.omega.ca/fr/sensors-and-sensing-equipment/pressure-and-strain/strain-gauges/p/SGD-LINEAR1-AXIS

9. ST. (2018). RM0351 Reference manual STM32L4x5 and STM32L4x6 advanced Arm®-based 32-bit MCUs.

10. ST. (2020). Description of STM32L4/L4+ HAL and low-layer drivers.

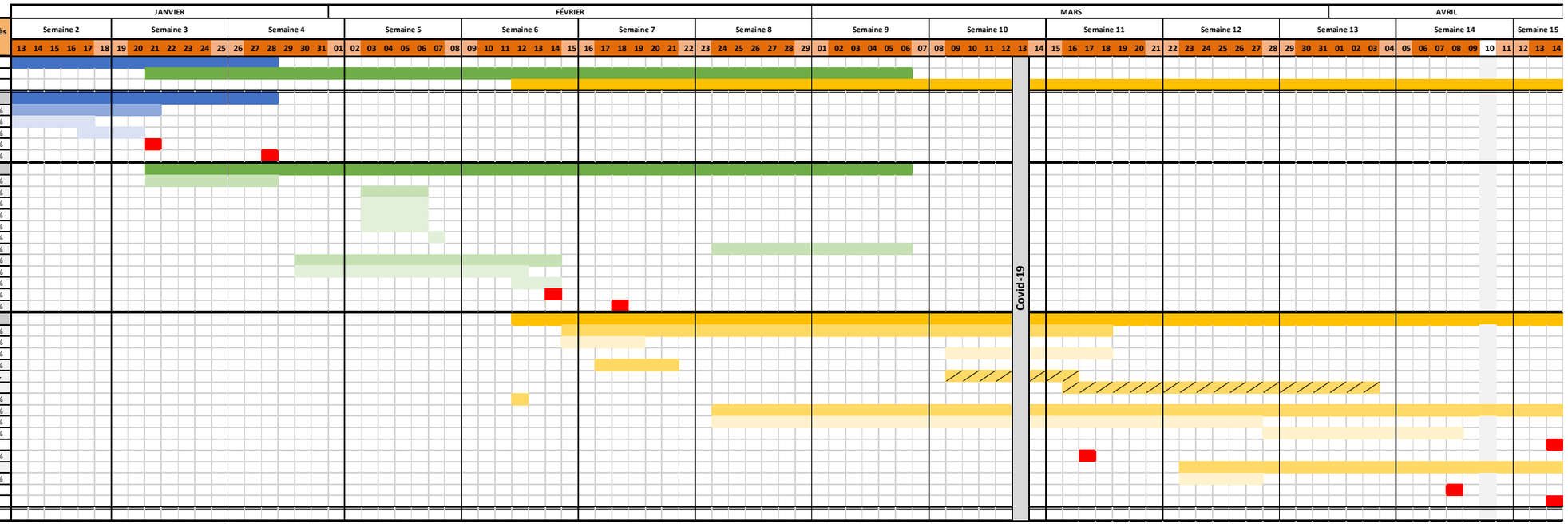
Annexe 1 – Diagramme de Gantt

Diagramme de Gantt **ELE795 - Omer 12 - Capteur de couple**

OMER
ÉQUIPE MARINE À PROPULSION HUMAINE

Dernière modification Date: 10-avr

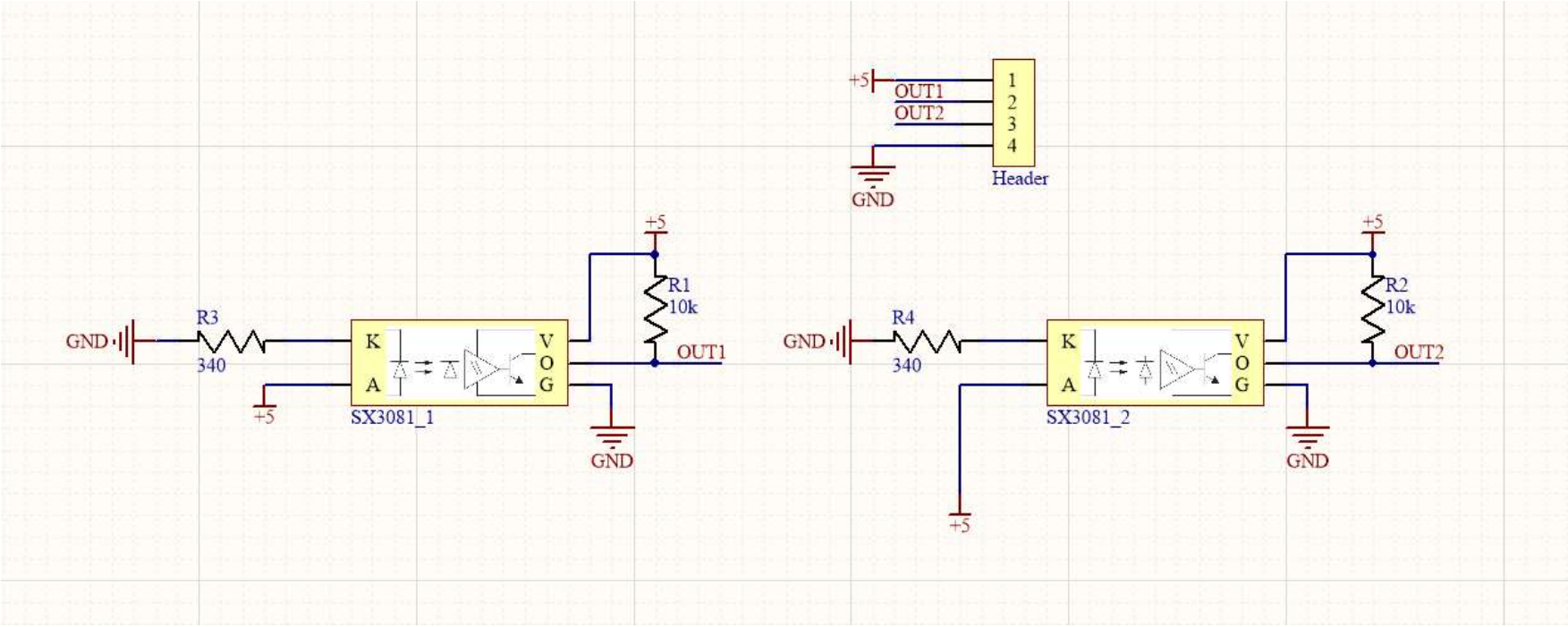
No.	Tâches	Durée (jours)	Date de début	Date de fin	Progrès %
1	Définition du projet	16	13-janv	28-janv	
2	Conception du capteur	28	21-janv	6-mars	
3	Réalisation du prototype et test	35	12-févr	14-avr	
1	**Définition du projet**	**16**	**13-janv**	**28-janv**	
1.1	Rédaction du rapport de définition du projet	**9**	13-janv	21-janv	100%
1.1.1	Rédaction	**5**	13-janv	17-janv	100%
1.1.2	Revision et correction	**4**	17-janv	20-janv	100%
1.1.3	Remise			21-janv	100%
1.2	Rencontre avec superviseure			28-janv	100%
2	**Conception du capteur**	**46**	**21-janv**	**6-mars**	
2.1	Évaluation des solutions	**8**	21-janv	28-janv	100%
2.2	Choix des pièces électroniques	**4**	3-févr	6-févr	100%
2.2.1	Capteur de couple	**4**	3-févr	6-févr	100%
2.2.2	Micro-processeur	**4**	3-févr	6-févr	100%
2.2.3	Communication avec le système embarqué d'Omer 12	**4**	3-févr	6-févr	100%
2.2.4	Commande des pièces	**1**	7-févr	7-févr	100%
2.3	Conception du banc de test	**12**	24-févr	6-mars	100%
2.4	Rédaction du rapport d'étape	**16**	30-janv	14-févr	100%
2.4.1	Rédaction	**14**	30-janv	12-févr	100%
2.4.2	Revision et correction	**3**	12-févr	14-févr	100%
2.4.3	Remise			14-févr	100%
2.5	Rencontre avec superviseure			18-févr	100%
3	**Réalisation du prototype et test**	**63**	**12-févr**	**14-avr**	
3.1	Programmation	**33**	15-févr	18-mars	100%
3.1.1	Familiarisation avec le logiciel	**5**	15-févr	19-févr	100%
3.1.2	Programmation d'essai	**10**	9-mars	18-mars	100%
3.2	Assemblage mécanique du capteur	**5**	17-févr	21-févr	100%
~~3.3~~	~~Assemblage du banc de test~~	~~**8**~~	~~9-mars~~	~~16-mars~~	~~33%~~
~~3.4~~	~~Test et ajustement~~	~~**10**~~	~~16-mars~~	~~2-avr~~	
3.5	Budget solutions retenues	**1**	12-févr	12-févr	100%
3.6	Rédaction du rapport de définition du projet	**51**	24-févr	14-avr	100%
3.6.1	Rédaction	**33**	24-févr	27-mars	100%
3.6.2	Revision et correction	**12**	28-mars	8-avr	100%
3.6.3	Remise			14-avr	
3.7	Rencontre avec superviseure			17-mars	100%
3.8	Présentation		23-mars	14-avr	
3.8.1	Monter le PPT		23-mars	27-mars	100%
3.8.2	Rencontre avec superviseure			8-avr	
3.8.3	Présentation			14-avr	



Annexe 2 – Chiffrier de calculs de résolution

Chiffrier de calculs de précision du timer						
Constantes				Calculs		
Microcontrôleur				Angle de déformation nominal	Degré	0,045961
Horloge	MHz	80				
Système	Bits	32		Temps pour une déformation pour 1 degré	s/Degré	0,001667
Timers	Bits	32		Temps pour un torque nominal	s	7,66011E-05
Arbre						
Diamètre extérieur	po	0,630		Compte pour entre deux pulses des dents	Coup	266666666,7
Diamètre intérieure	po	0,000		Compte du timer par degré	Coup	133333
Longueur	po	4,000		Compte du timer pour un torque nominal	Coup	6128
Nbr dents		2		Taille de timer		4294967295
Régidité du matériel	GPa	68,9				
Parmètres nominaux de course				Précision d'un coup de timer	Degré	0,0000075
Vitesse nominale	RPM	100		Précision d'un coup de timer	Nm	**0,00057**
Torque nominal	Nm	3,5				

Annexe 3 – Schéma du circuit imprimé



C

Annexe 4 – Fichier main.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
#include "t_torque.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

// Gestion des timers
#define INPUT_1 0
#define INPUT_2 1

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim2;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

// Gestion du timer
volatile uint8_t flag_edge1 =0;     // Indication du 1er front montant
volatile uint8_t flag_edge2 =0;     // Indication du 2e front montant
volatile uint32_t tim_capture[2];   // Valeur des captures des timers

// Données
volatile t_torque data_test;                // Données en cours de traitement
t_torque* data = &data_test;

// Communication série
uint8_t buffer[50];         // Tampon de caractères pour la transmission série
char WELCOME_MSG[]="OMER12 Capteur de couple\r\n";


/* USER CODE END PV */
```

```c
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */


/* USER CODE END PFP */


/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */


/* USER CODE END 0 */


/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */


  /* USER CODE END 1 */


  /* MCU Configuration--------------------------------------------------------*/


  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();


  /* USER CODE BEGIN Init */
  // Initialiser la donnée à zéro
  init_data(data);
  data->state = STATE_WAIT_EDGE1;
  data->state = 0;
  /* USER CODE END Init */


  /* Configure the system clock */
  SystemClock_Config();


  /* USER CODE BEGIN SysInit */


  /* USER CODE END SysInit */


  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  // Activer l'entrée 1 du timer 2
  HAL_TIM_IC_Start_IT(&htim2,   TIM_CHANNEL_1);
  // Activer l'entrée 2 du timer 2
  HAL_TIM_IC_Start_IT(&htim2,   TIM_CHANNEL_2);
  //Message de bienvenu
  HAL_UART_Transmit(&huart2,    (uint8_t*)WELCOME_MSG,  strlen(WELCOME_MSG),    HAL_MAX_DELAY);
  init_data(data);
  /* USER CODE END 2 */



  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {

 /* MACHINE D'ÉTATS */
      switch (data->state){

/* ÉTAT STATE_WAIT_EDGE1 */
      // En attente du 1er front montant
      case STATE_WAIT_EDGE1:
             // Drapeau du 1er front montant et une seule valeur capturée
             if (flag_edge1){

                 // L'entrée 1 a été activée avant la 2e
                 if (tim_capture[INPUT_1]!=0 && tim_capture[INPUT_2]==0){
```

E

```
                    // Mettre la valeur du timer dans la donnée pour la vitesse
                    data->raw_speed = tim_capture[INPUT_1];
                    // Vers le prochain état
                    data->state = STATE_SPEED;
                }
                // L'entrée 2 a été activée avant la 1er
                else if (tim_capture[INPUT_1]==0 && tim_capture[INPUT_2]!=0){
                    // Mettre la valeur du timer dans la donnée pour la vitesse
                    data->raw_speed = tim_capture[INPUT_2];
                   // Erreur: l'arbre trourne de sens inverse
                    data->state = ERR_EDGE_POS;
                }
                // Erreur
                else{
                    data->state = ERR_TMR_OVR1;
                }
            }
        break;

/* ÉTAT STATE_SPEED */
        // Calcul de la vitesse
        case STATE_SPEED:
            // Calcul de vitesse
            compute_speed(data);

            //Vérification d'erreur du calcul de vitesse
            if(data->speed_degree > ERR_SPEED_OVR_VALUE){
                // Erreur: La vitesse est physiquement impossible
                data->state = ERR_SPEED_OVR;
            }
            else if(data->speed_degree < 0){
                // Erreur: La vitesse est négative
                data->state = ERR_SPEED_NEG;
            }
            // Aucune erreur
            else{
                // Vers le prochain état
                data->state = STATE_WAIT_EDGE2;
            }
        break;

/* ÉTAT STATE_WAIT_EDGE2 */
        // En attente du 2e front montant
        case STATE_WAIT_EDGE2:
               // Drapeau du 2e front montant et avec 2 valeurs capturées
               if (flag_edge2){

                  // Les deux valeurs ont été capturées
                  if (tim_capture[INPUT_1]!=0 && tim_capture[INPUT_2]!=0){
                      // Mettre la valeur du timer dans la donnée pour l'angle
                      data->raw_angle = tim_capture[INPUT_2];
                      // Vers le prochain état
                      data->state = STATE_TORQUE;
                  }
                  // Erreur
                  else{
                       data->state = ERR_TMR_OVR2;
                  }
              }
        break;

/* ÉTAT STATE_TORQUE */
        // Calcul de l'angle et du torque
        case STATE_TORQUE:
              // Calcul de l'angle
              compute_angle(data);
              // Calcul du couple
              compute_torque(data);

              //Vérification d'erreur de l'angle
              if(data->angle > ERR_ANGLE_OVR_VALUE){
                  // Erreur: L'angle est physiquement impossible
                  data->state = ERR_ANGLE_OVR;
                  break;
              }

              //Vérification d'erreur du calcul du torque
              if(data->torque > ERR_TORQUE_OVR_VALUE){
                  // Erreur: Le couple est physiquement impossible
                  //data->state = ERR_TORQUE_OVR;
                  //break;
              }
```

```
            else if(data->torque < 0){
                // Erreur: Le couple est négative, le pilote force contre l'autre
                data->state = ERR_TORQUE_NEG;
                break;
            }

             // Aucune erreur
             // Vers le prochain état
             data->state = STATE_SEND_DATA;
      break;

/* ÉTAT STATE_SEND_DATA */
      // Calculs complétés, donnée à transmettre
      case STATE_SEND_DATA:

             // Former la string
            sprintf((char*)buffer, "%lu;%lu;%lu;%.1f;%.6f;%.2f;%d\r\n",
                    data->data_id,
                    data->raw_speed,
                    data->raw_angle,
                    data->speed_degree,
                    data->angle,
                    data->torque,
                    data->state);
            // Envoyer la string
            HAL_UART_Transmit(&huart2,  buffer, strlen((char*)buffer), HAL_MAX_DELAY);
            // Vers le prochain état
            data->state = STATE_RESET_DATA;
      break;

/* ÉTAT STATE_RESET_DATA */
      //Remise à zéro d'une donnée
      case STATE_RESET_DATA:

            // RAZ des drapeaux et des valeurs de capture
            flag_edge1 = 0;
            flag_edge2 = 0;
            tim_capture[INPUT_1] = 0;
            tim_capture[INPUT_2] = 0;

            // Remettre les champs de la donnée à zéro
            init_data(data);

            // Incrémenter l'identifiant de la données
            data->data_id ++;

            // Vers le prochain état
            data->state = STATE_WAIT_EDGE1;
      break;

/* ÉTAT ERREURS */
      // Gestion des erreurs
      default:
             // Former la string
            sprintf((char*)buffer, "%d\r\n",data->state);
            // Envoyer la string
            HAL_UART_Transmit(&huart2,  buffer, strlen((char*)buffer), HAL_MAX_DELAY);
            // Vers le prochain état
            data->state = STATE_RESET_DATA;
      break;
    }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
```

```
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 10;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;


  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART2;
  PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure the main internal regulator output voltage
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
  {
    Error_Handler();
  }
}


/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{


  /* USER CODE BEGIN TIM2_Init 0 */


  /* USER CODE END TIM2_Init 0 */


  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_IC_InitTypeDef sConfigIC = {0};


  /* USER CODE BEGIN TIM2_Init 1 */


  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 0xffffffff;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_FALLING;
  sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
  sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
  sConfigIC.ICFilter = 4;
  if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
```

```
  {
    Error_Handler();
  }
  if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
```

```c
}

/* USER CODE BEGIN 4 */

// Interrupt du mode input capture
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {

    //Timer 2
    if (htim == &htim2) {

        // Channel 1 PA0
        if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1){
            // Lire la valeur de capture du timer
            tim_capture[INPUT_1] = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_1);
        }
        // Channel 2 PA1
        else if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2){
            // Lire la valeur de capture du timer
            tim_capture[INPUT_2] = HAL_TIM_ReadCapturedValue(htim, TIM_CHANNEL_2);
        }

        // C'est le 1er front montant
        if (!flag_edge1 && !flag_edge2){
            // Mettre le timer à zéro
            __HAL_TIM_SET_COUNTER(htim, 0);
            // Faire flasher la led verte
            HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
            // Mettre le drapeau du premier front montant à 1
            flag_edge1 = 1;
        }   // Si c'est le deuxième front montant
        else{
            // Mettre le drapeau du 2e front montant à 1
            flag_edge2 = 1;
        }
    }
    // Les drapeaux flag_edge sont remis à zéros dans la boucle principale
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */

  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(char *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Annexe 5 – Fichier t_torque.h

```
/**
******************************************************************************
  * @file           : t_torque.h
  * @brief          : Header for t_torque.c file.
  *                   Calcul du couple d'un arbre
  ******************************************************************************
  * Auteur: Charles-Éric Lepage
  * Date: 2020-03
  *
  * Cours: ELE795-HIV20
  * Projet: Système de mesure de couple d'Omer 12
  *
  ******************************************************************************
  */
#ifndef __T_TORQUE_
#define __T_TORQUE_

#include <stdio.h>
#include <string.h>
#include "stm32l4xx_hal.h"
#include <math.h>

/*******************************************************************************
 *                      DEFINITIONS DES TYPES ET DE CONSTANTES
 *******************************************************************************/

// Constantes physiques
#define SHAFT_L 0.1         // Longueur de l'arbre entre les deux capteurs en m
#define SHAFT_G 68.9        // Constante d'élasticité du matériel en GPa (Aluminium=68.9)
#define SHAFT_DE 0.016      // Diamètre extérieur de l'arbre en m
#define SHAFT_DI 0.0        // Diamètre intérieur de l'arbre en m
#define NUM_TEETH 2         // Nombre de dents par tour de l'arbre
#define DELTA_T 0           // Constant de temps entre les deux capteurs pour à 0Nm en coups d'horloge

// Status des données pour la machine d'états
// Note: la dizaine du code d'erreur représente l'état ou elle a été générée
#define STATE_WAIT_EDGE1 0  // En attente du 1er front montant
#define ERR_TMR_OVR1    -1  // Timer dépassé en attendant le 1er front
#define ERR_EDGE_POS    -2  // Le deuxième front montant est arrivé avant le premier

#define STATE_SPEED      1  // Calculs de la vitesse
#define ERR_SPEED_OVR   -11 // Valeur de la vitesse trop grande
#define ERR_SPEED_OVR_VALUE 720 // Valeur de la vitesse considérée trop grande en degré par sec
#define ERR_SPEED_NEG   -12 // Valeur de la vitesse négative (fesant reculer le sous-marin)

#define STATE_WAIT_EDGE2 2  // En attente du 2e front montant
#define ERR_TMR_OVR2    -21 // Timer dépassé en attendant le 2er front

#define STATE_TORQUE     3  // Calculs de l'angle de déformation et du couple
#define ERR_ANGLE_OVR   -31 // Valeur de l'angle trop grande
#define ERR_ANGLE_OVR_VALUE 1.0 // Valeur de l'angle considérée trop grande
#define ERR_TORQUE_OVR  -33 // Valeur du couple trop grande
#define ERR_TORQUE_OVR_VALUE 1000 // Valeur du couple considérée trop grande en Nm
#define ERR_TORQUE_NEG  -34 // Valeur du couple négative (fesant reculer le sous-marin)

#define STATE_SEND_DATA  4  // Donnée envoyée sur le port série

#define STATE_RESET_DATA 5 // Remise à zéro d'une donnée


/*******************************************************************************
 *                                   STRUCTURE
 *******************************************************************************/

// Données d'une prise de mesure du torque
typedef struct{

    uint32_t data_id;       // Numérotation séquenciel de la mesure du couple
    uint32_t raw_speed;     // Valeur brut du timer pour la vitesse
    uint32_t raw_angle;     // Valeur brut du timer pour l'angle de déformation
    float speed_degree;     // Vitesse de rotation de l'arbre en degrée par sec
    float angle;            // Angle de déformation de l'arbre
    float torque;           // Torque en Nm
    int8_t state;           // Status de la donnée
```

```
}t_torque;


/*****************************************************************************
 *                        DECLARATIONS DES FONCTIONS PUBLIQUES
 *****************************************************************************/

/*
    INIT_DATA
    Cette fonction permet d'initialiser une variable de type t_torque
    Paramètres :
         data: donnée à initialiser à zéro
    Retour :
*/
void init_data(t_torque* data);


/*
    COMPUTE_SPEED
    Cette fonction permet de calculer la vitesse de rotation en degré/seconde
    Paramètres :
         data: donnée contennant la valeur brut du timer pour le calcul de vitesse
    Retour :
*/
void compute_speed(t_torque* data);


/*
    COMPUTE_ANGLE
    Cette fonction permet de calculer l'angle de déformation en degré
    Paramètres :
         data: donnée contennant la vitesse et la valeur brut du timer pour l'angle
    Retour :
*/
void compute_angle(t_torque* data);


/*
    COMPUTE_TORQUE
    Cette fonction permet de calculer le couple
    Paramètres :
         data: donnée contennant l'angle de déformation
    Retour :
*/
void compute_torque(t_torque* data);

#endif
```

Annexe 6 – Fichier t_torque.c

```
/**
******************************************************************************
  * @file           : t_torque.c
  * @brief          : Calcul du couple d'un arbre
  ******************************************************************************
*
  * Auteur: Charles-Éric Lepage
  * Date: 2020-03
  *
  * Cours: ELE795-HIV20
  * Projet: Système de mesure de couple d'Omer 12
  ******************************************************************************
*
  */
#include "t_torque.h"

/******************************************************************************
*
 *                      DEFINITIONS DES FONCTIONS PRIVÉES
 ******************************************************************************
*/



/******************************************************************************
*
 *                      DEFINITIONS DES FONCTIONS PUBLIQUES
 ******************************************************************************
*/

/*
    INIT_DATA
    Cette fonction permet d'initialiser une variable de type t_torque
    L'identifiant data_id et l'état state sont gérés par le main
    Paramètres :
         data: donnée à initialiser à zéro
    Retour :
*/
void init_data(t_torque* data){

    data->raw_speed = 0;
    data->raw_angle = 0;
    data->speed_degree = 0.0;
    data->angle = 0.0;
    data->torque = 0.0;
}

/*
    COMPUTE_SPEED
    Cette fonction permet de calculer la vitesse de rotation en degrée/seconde
    Paramètres :
         data: donnée contennant la valeur brut du timer pour le calcul de vitesse
    Retour :
*/
void compute_speed(t_torque* data){

    /* Équation:
     * v = (360degrés/Nbr de dents sur l'arbre) * (Valeur du timer / Vitesse de
l'horloge)
     */
    data->speed_degree    =    ((360/NUM_TEETH)    *    (float)data->raw_speed    /
SystemCoreClock);
}


/*
```

```
       COMPUTE_ANGLE
       Cette fonction permet de calculer l'angle de déformation en degré
       Paramètres :
            data: donnée contennant la vitesse et la valeur brut du timer pour
l'angle
       Retour :
   */
   void compute_angle(t_torque* data){

       /* Équation:
        * angle = (Valeur du timer / fréquence de l'horloge) * vitesse en degré par
sec
        */
       data->angle =  (((float)data->raw_angle - DELTA_T / SystemCoreClock) * data-
>speed_degree;

   }


   /*
       COMPUTE_TORQUE
       Cette fonction permet de calculer le couple
       Paramètres :
            data: donnée contennant l'angle de déformation
       Retour :
   */
   void compute_torque(t_torque* data){

       /* Équation moment d'inertie:
        * J = pi/32 * (diamètre extérieur^4 - diamètre intérieur^4
        * Équation couple:
        * torque = angle * moment d'inertie * module d'élasticité / longueur de
l'arbre
        */
       float j = M_PI / 32 * (powf(SHAFT_DE,4) - powf(SHAFT_DI,4));
       data->torque = data->angle  * j * SHAFT_G * (powf(10,9)) / SHAFT_L;

   }
```

oiq.qc.ca