
RAPPORT DE PROJET

Projet d'électronique n°4 : Le contrôleur de moteur DC

Auteurs :

Ibrahim
Thomas
Hugo
Quentin

KEBE
DUCLOS
VIDAL
FEVE

Enseignant :

M. LOPES

Ce quatrième projet d'électronique a été très différent des trois précédents car il était en relation direct avec notre projet annuel "ECE CUP". Ce projet s'est étalé sur environ un mois et demi. En effet, nous devions mettre en place de nouvelles fonctionnalités pour notre robot et ceci à l'aide de nos connaissances acquises en "Electronique fondamentale". Nous devions donc appliquer tout ce que nous avons appris dans le but de réussir les tâches qui nous étaient demandées dans le sujet. Nous avons pour objectif de développer les fonctionnalités suivantes :

- Le robot doit pouvoir réguler sa vitesse
- Le robot doit être capable de se stabiliser
- Le robot doit pouvoir communiquer en Wi-Fi avec un ordinateur
- Mettre en place un site pour contrôler et initialiser les différents paramètres du robot

Nous attestons que ce travail est original, qu'il est le fruit d'un travail commun au binôme et qu'il a été rédigé de manière autonome.

Paris, le 27/03/2022

Table des matières

I. Objectif.....	3
II. Glossaire	3
III. L'équipe.....	4
A. Présentation de l'équipe.....	4
B. Organisation de l'équipe.....	4
C. Diagramme de GANTT	5
IV. Contexte et problématique	5
A. Contexte	5
B. Problématique	6
V. Conception	6
A. Architecture fonctionnelle	6
B. Architecture matérielle	7
C. Architecture logicielle	8
VI. Développement	9
A. Module 1 : Le contrôleur de moteur DC	9
B. Module 2	10
C. Module 3 : Correction PID	16
VII. Tests et validation	18
A. Module 1.....	18
B. Module 2.....	18
C. Module 3	22
VIII. Bilan	24
A. État d'avancement	24
B. Pertinence de la solution technique.....	24
C. Bilan sur le travail d'équipe.....	24
IX. Sources	25
X. Annexes.....	26

I. Objectif

Ce rapport a pour but de résumer et expliquer comment et par quels moyens nous avons pu accomplir ou non les missions qui nous étaient demandées. Ce document retrace notre avancée et expose les problèmes que nous avons pu rencontrer et les solutions trouvées afin de résoudre ces derniers et mener à bien notre projet.

Dans ce rapport se trouve aussi les ressentis de tous, nos avis sur notre travail d'équipe, notre organisation, notre communication.

II. Glossaire

Acronyme	Signification	Explication
PID	Proportionnel, intégral, dérivé	Système de contrôle permettant d'améliorer les performances d'un asservissement.
SSID-Wifi	Service Set Identifier	Nom d'un réseau wifi, composé au maximum de 32 caractères alphanumériques.
HTML	HyperText Markup Language	Langage utilisé afin de créer et représenter le contenu d'une page web et sa structure.
PWD	Print Working Directory	Commande qui affiche le nom du répertoire courant.
MDNS	Multicast Domain Name System	Dans les réseaux informatiques, le protocole DNS multidiffusion résout les noms d'hôte en adresses IP au sein de petits réseaux qui n'incluent pas de serveur de noms local.
WIFI	Wireless Fidelity	Technique qui permet la communication sans fil entre divers appareils
IP	Internet Protocol	L'IP est un protocole de communication pour le réseau Internet

III. L'équipe

A. Présentation de l'équipe

Nous sommes la SQUAD1063, composé de quatre étudiants d'ING2 du TD6. L'année dernière nous étions trois et ce n'est qu'en début d'année que Quentin issu de PREPAC nous a rejoint. Quentin a su s'adapter et s'investir directement dans le projet ECE CUP il a donc été d'une aide importante malgré son manque d'expérience en électronique dû à la prépa accélérée.

Notre groupe est soudé, nous avons une bonne communication et une bonne harmonie ce qui nous a permis d'avancer tous ensemble dans la même direction et mener à bien notre objectif commun.



Figure 1 - Photo de groupe

B. Organisation de l'équipe

Notre équipe est assez bien organisée, nous essayons de programmer des suivis de projet assez souvent environ un par semaine, en plus des heures d'autonomie ECE CUP prévus dans notre emploi du temps. Nous nous sommes réparti le travail de manière simple, notre Trello (qui existe depuis le début de l'année) nous a vraiment aidé à nous repérer dans l'avancement du projet. De plus, il règne une bonne ambiance au sein de notre équipe ce qui permet une communication fluide entre nous et une organisation efficace. La force de notre équipe réside dans les compétences de tout un chacun. Thomas a apporté ses connaissances en codage

et les a mises au service du groupe ce qui a été très bénéfique à la réalisation de ce projet. Ibrahim, a lui mis ses compétences en électronique en avant pour produire ce projet. Hugo et Quentin ont aidés Ibrahim et Thomas dans leurs tâches en fonction de leurs compétences, ils ont également de bonnes capacités dans les domaines cités plus tôt. Ils se sont également servis de leurs qualités rédactionnelles afin de créer le meilleur rapport possible.

C. Diagramme de GANTT



Figure 2 – Diagramme de Gantt

IV. Contexte et problématique

A. Contexte

Le gyropode est une invention de Dean KAMEN, Susan D DASTOUS, Robert DUAGAN et Michael GUAY. Le gyropode a été inventé en l'an 2000 et son premier modèle fût le SEGWAY TPI167. Ce premier modèle a été précurseur du gyropode à 6 roues inventé en avril 2003 pour les handicapés. Vente après-vente, en 2006 la technologie leensteer qui permet de faire tourner le gyropode grâce au poids du corps et aujourd'hui on s'en sert dans la vie de tous les jours pour nos déplacements quotidiens.

Dans notre projet le gyropode sera utile pour stabiliser notre robot afin d'avoir des déplacements plus précis.

B. Problématique

L'objectif de ce projet est de pouvoir stabiliser le robot et de le contrôler via une interface web. Sur cette interface on doit pouvoir régler la vitesse des moteurs afin de diriger le robot dans ses déplacements. Il faut donc rendre les moteurs plus précis et pour se faire il est nécessaire d'intégrer un PID il faut donc pouvoir le régler aussi à l'aide du site.

Comment régler la précision des moteurs pour rendre le robot parfaitement stable ?

V. Conception

A. Architecture fonctionnelle

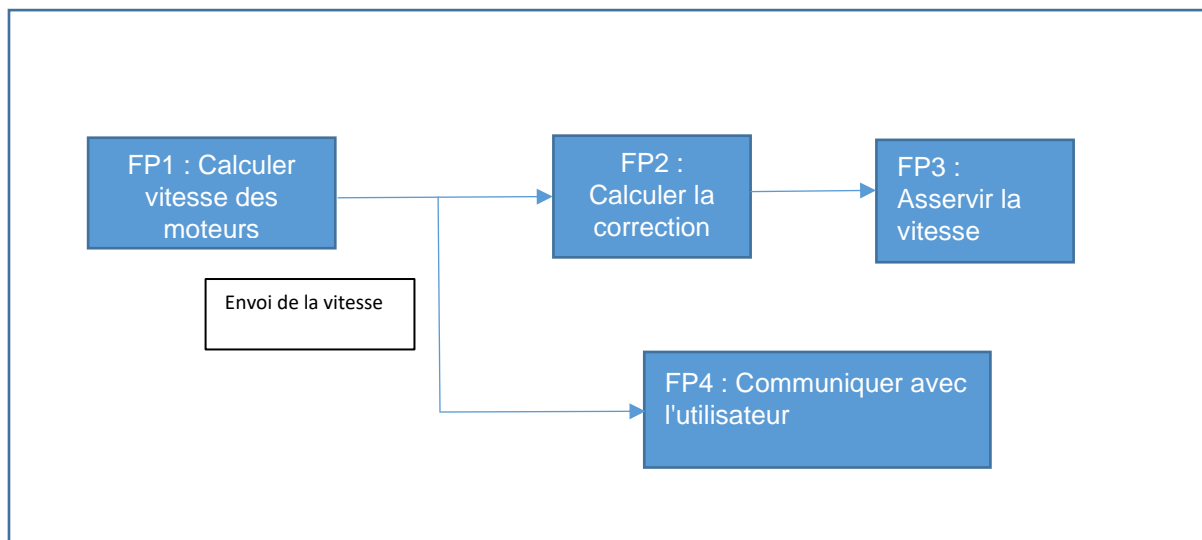


Figure 3 – Architecture fonctionnelle

Ci-dessus l'architecture fonctionnelle de notre projet, nous avons pour objectif de d'abord réaliser la Fonction Principale 1 qui consiste à calculer la vitesse des moteurs puis avec cette vitesse calculée nous pouvons réaliser FP2 c'est à dire calculer la correction. Suite à cela nous pouvons asservir la vitesse. Toutes ces fonctions sont réalisées tout en communiquant avec l'utilisateur.

B. Architecture matérielle

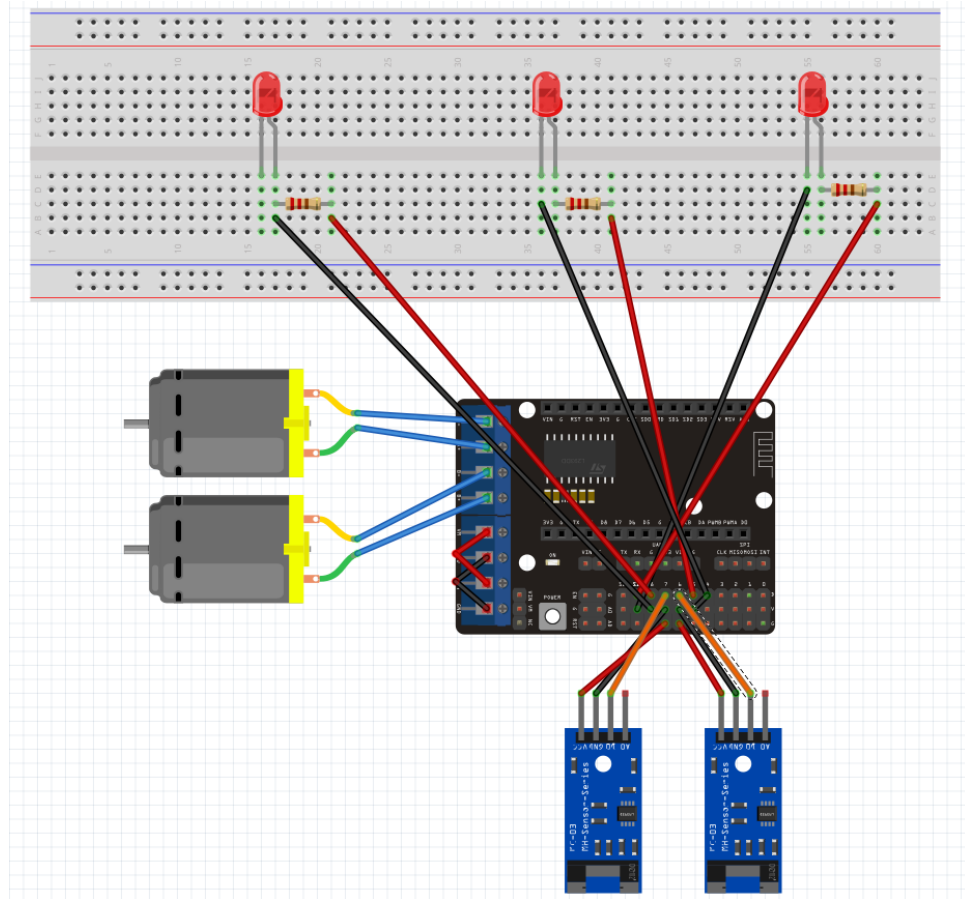


Figure 4 - Schéma de branchement

Les composants utilisés pour le projet sont les suivants :

- 2 capteurs pour mesurer la vitesse des roues codeuses
- 2 moteurs DC pour faire avancer le robot
- 1 esp8266
- 3 LED pour indiquer
- 3 résistances de 220Ω .

C. Architecture logicielle

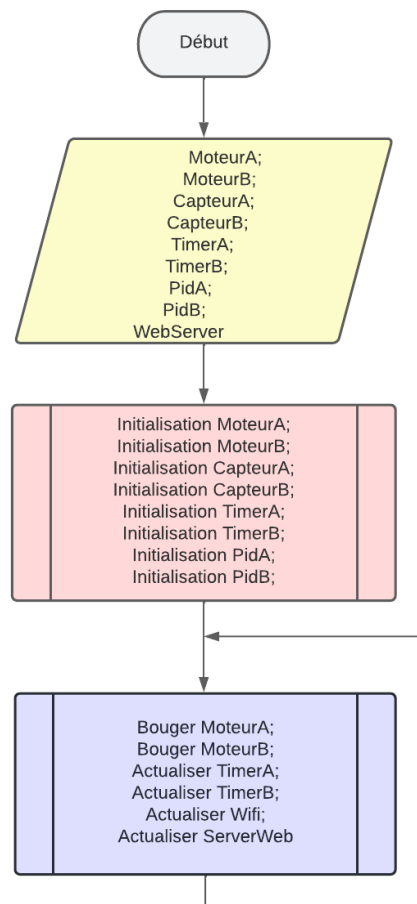


Figure 5 – Algorithme du code complet

L'algorithme ci-dessus, représente l'architecture logicielle globale de notre code. Dans un premier temps, toutes les variables sont déclarées au début du programme.

Par la suite elles sont toutes initialisées dans le setup qui est représenté par le processus en rouge.

Et enfin elles sont actualisées à chaque tour de boucle de la fonction loop. L'actualisation de chacune des données est précisée dans la partie développement de chaque module.

VI. Développement

Ce projet est divisé en quatre modules différents. En effet, pour mener à bien notre projet nous devons réaliser les parties les unes après les autres. Cette démarche nous a forcé à nous organiser, à communiquer et c'est pour cela que la répartition des tâches a été très importante.

Vous trouverez dans les paragraphes suivants la description de quatre différents modules et la manière avec laquelle nous avons procédé pour remplir toutes nos missions.

A. Module 1 : Le contrôleur de moteur DC

Ce module consiste en l'écriture d'un code permettant de contrôler les moteurs DC du robot. Pour ce faire, nous avons fixé les roues codeuses sur les moteurs et avons établi un code qui calcule la vitesse angulaire.

Ci-dessous nous pouvons retrouver l'organigramme de notre code Arduino.

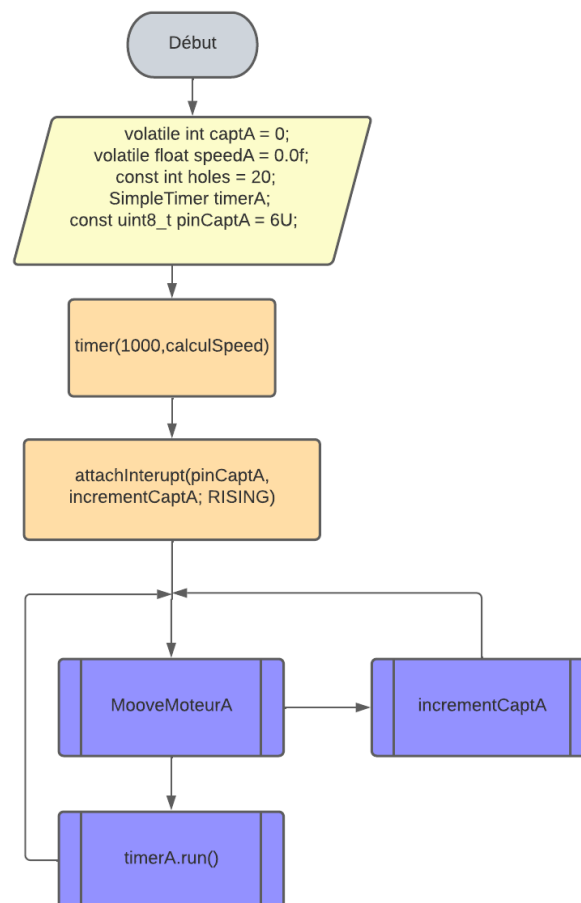


Figure 6 1- Algorithme calcul de la vitesse angulaire

Cet organigramme représente le calcul de la vitesse angulaire d'un moteur. Il suffira de le dupliquer pour le second moteur.

On commence ainsi par déclarer les variables nécessaires. On déclare ainsi les pins du moteur et du capteur, ainsi qu'un compteur pour stocker le nombre de trous de la roue codeuse qui auront été traversés lors de l'intervalle du timer. Dans la fonction setup on initialisera notre fonction d'interruption ainsi que notre chronomètre.

La fonction d'interruption permet donc d'augmenter le compteur de trous à chaque fois qu'un front montant est détecté sur la pin du capteur.

Le chronomètre quant à lui, nous permettra d'effectuer une tâche à chaque fois que son intervalle aura été écoulé.

Une fois l'intervalle du chronomètre finit celui-ci, il déclenche la fonction permettant de calculer la vitesse du moteur.

Cette vitesse est calculée en tour par seconde. Elle est donnée par la relation suivante :

$$\omega = \frac{\text{compteur}}{\text{nombre de trous}} = \frac{\text{compteur}}{20.0}.$$

On obtient ainsi une vitesse angulaire.

B. Module 2

Ce module consiste en l'écriture d'un code permettant d'établir une communication entre la carte ESP8266 et l'utilisateur. Pour ce faire nous avons utilisé le module Wifi de la carte ESP8266, ainsi qu'un Web Server et une Web Socket.

Le module Wifi permettra d'établir une connexion Wifi entre notre ESP8266 et le réseau local. Ainsi, l'ESP sera alors disponible sur le réseau local et pourra donc établir une communication avec un autre appareil du réseau local. Cependant la connexion wifi pose problème, il faut connaître l'adresse IP attribuée à la carte par le réseau Wifi et en cas de crash du routeur wifi, il sera impossible de continuer le programme.

Nous avons donc réussi à résoudre ce problème en utilisant la bibliothèque WiFiMulti pour permettre à la carte de se reconnecter à un Wifi déjà stocké dans sa mémoire en cas de perte de réseau sur le wifi actuel. De plus, l'IP étant en permanence modifiée, il faudrait alors à chaque fois se connecter avec la carte et observer l'adresse IP attribuée. Or ceci n'est pas pratique. Pour remédier à ceci, nous avons utilisé deux solutions. La première est la MDNS, notre carte est alors attribuée à un nom constant sur le réseau. Malheureusement, la MDNS n'est pas disponible pour

tous les appareils pouvant se connecter à l'ESP. Ainsi notre seconde solution a été de fixer l'adresse IP de la carte.

Le Web Server nous permettra d'envoyer une page HTML ainsi que les fichiers de styles CSS et les fichiers de code JavaScript à l'utilisateur. Sur cette page HTML on retrouvera des éléments permettant de prendre le contrôle des moteurs du robot.

La Web Socket nous permettra d'échanger des données en continu entre tous les utilisateurs utilisant la page Web.

La Web Socket pose cependant de gros soucis en termes de sécurité. Effectivement, toutes personnes ayant accès à la page Web contrôlant le robot pourrait alors modifier les paramètres et empêcher son bon fonctionnement. Ainsi pour remédier à ce problème nous avons utilisé une page de connexion demandant le SSID ainsi que le mot de passe à fournir pour accéder à la page de contrôle.

Cependant ceci n'est pas suffisant car toutes personnes possédant l'url peuvent alors accéder directement à la page Web. Pour ce faire, nous avons alors mis en place un système d'authentification permettant de créer un token et de l'enregistrer dans le navigateur web sous forme de cookie. Ce cookie sera par la suite demandé lors de la connexion à la Web Socket. Si celui-ci n'est pas présent cela signifie que la personne est une intruse.

Nous avons alors réalisé un organigramme simplifié du module ci-dessous :

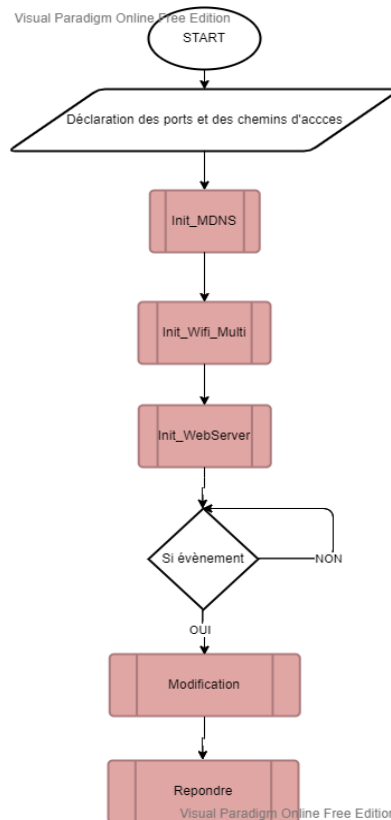
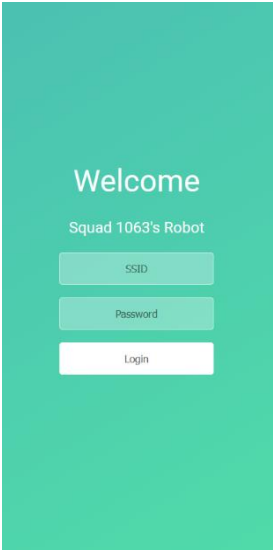


Figure2 7 - algorithme module 2

On peut observer qu'on commence tout d'abord par initialiser et créer toutes les connexions Wifi et Web Server.

Par la suite, le code dépend fortement des évènements survenant sur la Web Socket. Lors de la connexion d'un utilisateur au Web Server celui-ci lui renvoi la page html lui permettant de se connecter à la page de contrôle.



Welcome

Squad 1063's Robot

SSID

Password

Login

Figure 8 – Affichage du login de l'interface web

Si l'utilisateur rentre les bons identifiants, le Web Server le redirige vers la page de contrôle des moteurs.

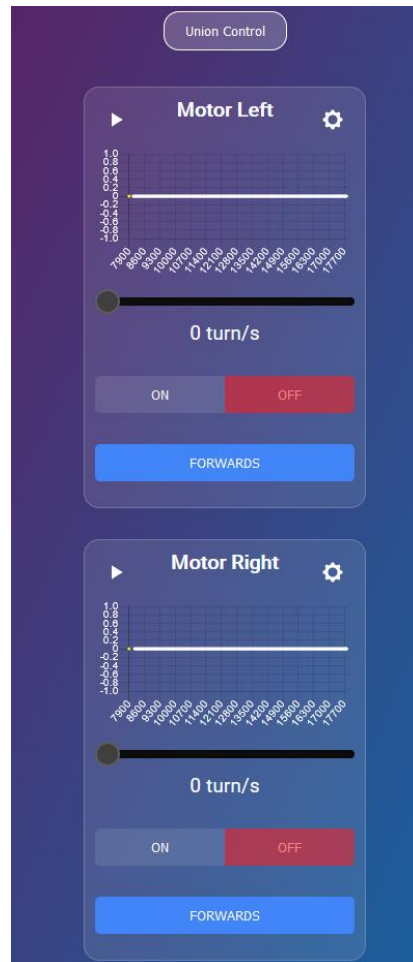


Figure 9 - Interface web moteur 2

L'utilisateur peut alors réaliser plusieurs actions, il peut commander l'allumage ou l'extinction de chaque moteur. Pour ce faire chaque bouton est relié à une fonction javascript qui une fois déclencher envoie un message via la Web Socket.

Ce message contient un objet JSON contenant un ou plusieurs attributs ainsi que leur valeur, cet objet JSON est tout d'abord transformé en une chaîne de caractères. Lorsque ce message est reçu sur la Web Socket, celle-ci le transforme en un objet JSON à nouveau à l'aide de la bibliothèque ArduinoJSON. Une fois le message transformé, en fonction des attributs présents on effectuera des modifications sur les différentes variables globales permettant de contrôler l'état des moteurs et des PID.

Enfin, la Web Socket notifiera tous ses clients pour qu'ils conservent tous le même état au niveau des commandes.

Il est également possible de modifier les deux moteurs en même temps en cliquant sur le bouton "Union Control". Ce bouton permet alors de cacher les deux moteurs et d'afficher un espace de contrôle permettant de contrôler les deux moteurs en même temps.

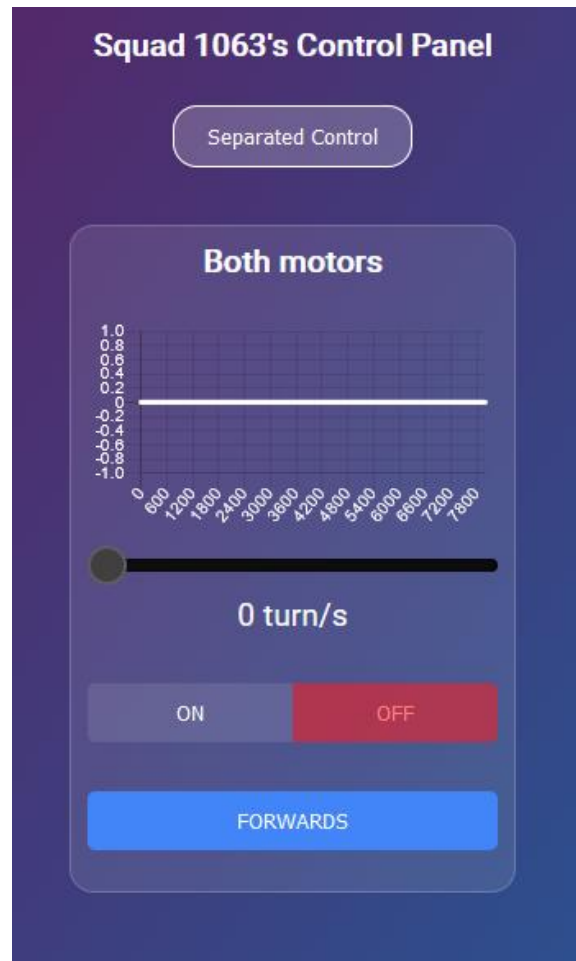


Figure 10 - Interface Web pour les deux moteurs

Il nous reste plus qu'à tracer les courbes représentatives correspondantes à la commande ainsi qu'à la vitesse calculée par les capteurs. La courbe de la commande sera tracée en blanc, tandis que la courbe de la vitesse sera tracée en jaune.

Pour tracer la courbe nous utilisons un module javascript Chart.js qui permet de tracer des graphiques. Pour modifier les graphiques, nous ajoutons des données toutes les 100ms dans des tableaux. Les données de la courbe de vitesse sont également transmises par Web Socket.



Figure 11 – Capture interface web des courbes de la vitesse et de la commande

C. Module 3 : Correction PID

Dans cette partie, il nous était demandé de mettre en place un PID pour corriger l'erreur induite de nos moteurs. Afin de pouvoir implémenter cette fonction, nous avons créé une classe dans laquelle on a comme variable, la consigne, la somme des erreurs et un delta erreur correspondant à la différence entre l'erreur actuelle et l'ancienne erreur.

Pour ce faire, il faut d'abord vérifier si le moteur est allumé ou pas avant d'attribuer des valeurs à nos variables car le correcteur ne peut être actif s'il n'y a aucune erreur. Une fois le moteur en marche on attribue la vitesse voulue comme consigne, on calcule l'erreur en faisant la différence entre la consigne et la vitesse actuelle du moteur, et on somme les erreurs à chaque tour de boucle.

Pour calculer la nouvelle consigne, on se sert de la formule suivante :

$$u(t) = K_p * \epsilon(t) + K_i * \sum \epsilon(t) + K_d * \int \epsilon(t)$$

Avec $\epsilon(t)$ l'erreur.

On a donc K_p qui multiplie l'erreur pour le régulateur P, K_i qui multiplie la somme de l'erreur pour le régulateur I et K_d qui multiplie l'intégrale de l'erreur (c'est-à-dire la différence entre l'erreur actuelle et l'erreur passée) pour le régulateur D.

Afin de réinitialiser nos valeurs pour ne pas avoir de valeurs immenses on les remet à 0 lorsque le moteur est éteint. Aussi la somme des erreurs est plafonnée à 2 pour éviter d'avoir des sommes trop grandes. Voici un algorithme de fonctionnement.

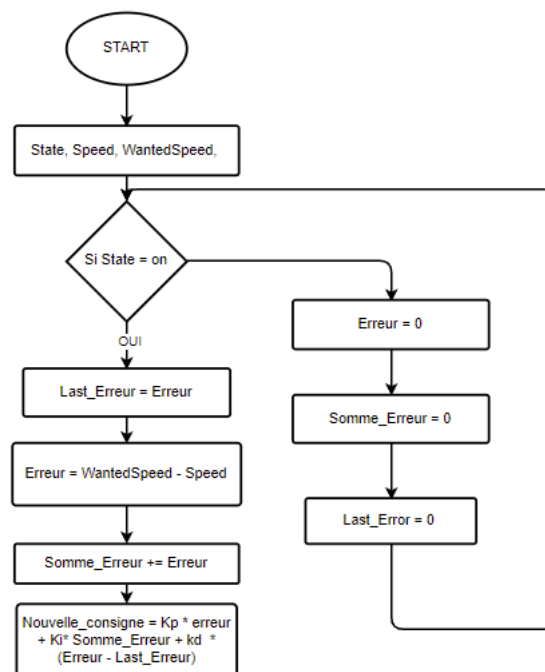


Figure 12 3- Algorithme du PID

On applique ce programme à chacun des moteurs individuellement en changeant la vitesse voulue et l'état.

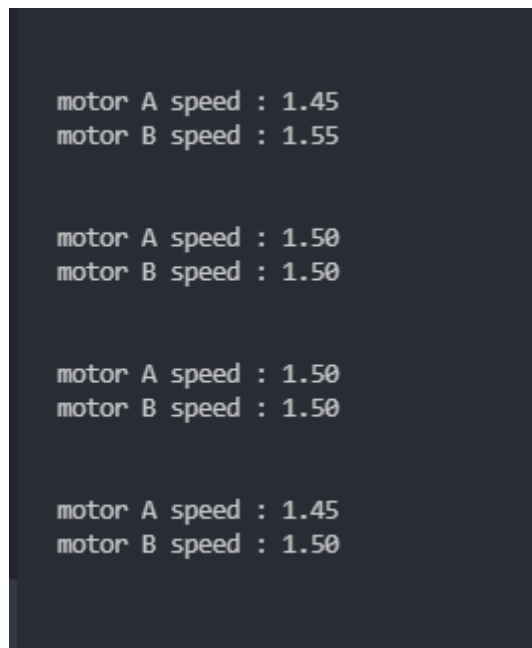
A noter, les réglages de K_p , K_i et K_d se font manuellement via l'interface Web. De plus, elles sont mises automatiquement à 0 au démarrage du programme.

Ce code peut alors être dupliqué sur l'autre moteur.

VII. Tests et validation

A. Module 1

La réussite du module 1 repose sur le bon fonctionnement des roues. En effet, nous sommes parvenus à mesurer la vitesse des roues de notre robot. L'affichage de la vitesse se fait simplement en console.



```
motor A speed : 1.45
motor B speed : 1.55

motor A speed : 1.50
motor B speed : 1.50

motor A speed : 1.50
motor B speed : 1.50

motor A speed : 1.45
motor B speed : 1.50
```

Figure 13 – Capture des mesures de vitesse

On peut donc affirmer que ce premier module est validé.

B. Module 2

Le module 2 consiste à créer un contrôle à distance des moteurs de notre robot et cela grâce à une interface web.

Pour cela nous avons d'abord vérifié que l'esp est bien connectée au wifi.

```
PS C:\Users\thoma> arp -a

Interface : 192.168.1.21 --- 0x6

Adresse Internet      Adresse physique      Type
192.168.1.1           30-93-bc-08-56-30     dynamique
192.168.1.10          30-24-78-9d-d9-79     dynamique
192.168.1.13          f8-4f-ad-a9-ac-9c     dynamique
192.168.1.18          fc-3f-db-e9-98-54     dynamique
192.168.1.19          48-d2-4f-bc-47-9a     dynamique
( 192.168.1.184        f4-cf-a2-ef-e5-02     dynamique )
192.168.1.255         ff-ff-ff-ff-ff-ff     statique
224.0.0.2             01-00-5e-00-00-02     statique
224.0.0.22            01-00-5e-00-00-16     statique
224.0.0.250           01-00-5e-00-00-fa     statique
224.0.0.251           01-00-5e-00-00-fb     statique
224.0.0.252           01-00-5e-00-00-fc     statique
232.0.3.32            01-00-5e-00-03-20     statique
232.0.3.35            01-00-5e-00-03-23     statique
232.0.10.87           01-00-5e-00-0a-57     statique
232.0.33.1            01-00-5e-00-21-01     statique
239.255.255.250       01-00-5e-7f-ff-fa     statique
255.255.255.255       ff-ff-ff-ff-ff-ff     statique
```

Figure 14 – Capture des différentes adresses IP

La commande “**arp-a**” entrée en console permet d'obtenir toutes les adresses IP présentes sur le réseau. Nous pouvons donc vérifier la présence ou non de l'esp sur le réseau. Cette vérification se fait en trouvant l'adresse IP “**192.168.1.184**” qui correspond à celle que nous avons affectée à l'ESP8266.

```
PS C:\Users\thoma> ping squad1063.local

Envoi d'une requête 'ping' sur squad1063.local [192.168.1.184] avec 32 octets de données :
Réponse de 192.168.1.184 : octets=32 temps=39 ms TTL=255
Réponse de 192.168.1.184 : octets=32 temps=55 ms TTL=255
Réponse de 192.168.1.184 : octets=32 temps=75 ms TTL=255
Réponse de 192.168.1.184 : octets=32 temps=80 ms TTL=255

Statistiques Ping pour 192.168.1.184:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 39ms, Maximum = 80ms, Moyenne = 62ms
```

Figure 15 – Capture de la réception des Paquets

La commande “**ping**” entrée en console permet d'envoyer des “Paquets” afin de tester si l'adresse arrive à recevoir ces “Paquets”. On indique en console si les “Paquets” ont bien été reçus, sur cette capture on voit que les quatre “Paquets” ont été reçus ce qui confirme que l'adresse est bel et bien accessible.

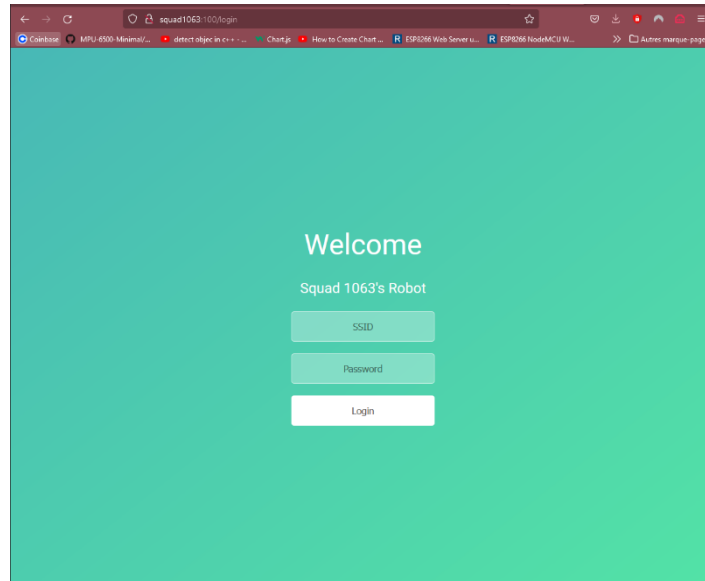


Figure 16 – Login de l'interface web

Sur la capture ci-dessus on voit bien que nous pouvons accéder au web server car on obtient bien la page html.

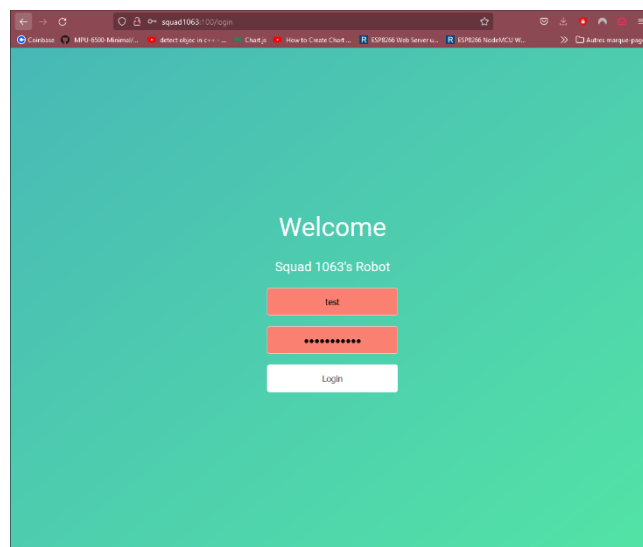


Figure 17 - Sécurité de l'interface web

Cette capture ci-dessus illustre la sécurité de notre page web, si l'utilisateur rentre les mauvais identifiants il n'est pas redirigé vers la page de contrôle.

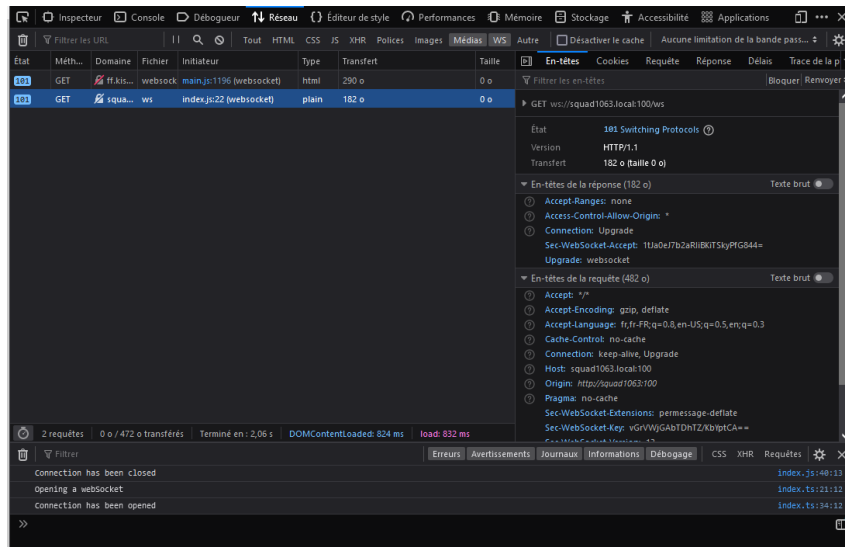


Figure 18 – Capture prouvant la création de la Web Socket

Cette capture permet de montrer que la Web Socket est bien créée lors du chargement de la page web. On peut notamment y observer les messages dans la console qui montre la connexion à la web socket de l'ESP8266 ainsi que la présence d'une web socket dans le gestionnaire réseau de la page web.

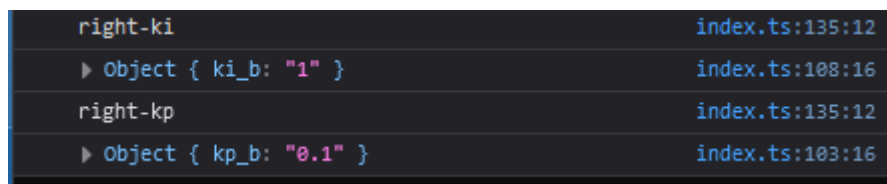


Figure 19 – Capture illustrant le protocole de communication entre les deux sockets

Cette capture d'écran permet de mettre en évidence le protocole de communication entre les deux web sockets.

On peut donc affirmer que ce second module est validé.

C. Module 3

Le module 3 consiste à créer un asservissement en vitesse d'un des deux moteurs et cela à l'aide d'un PID.

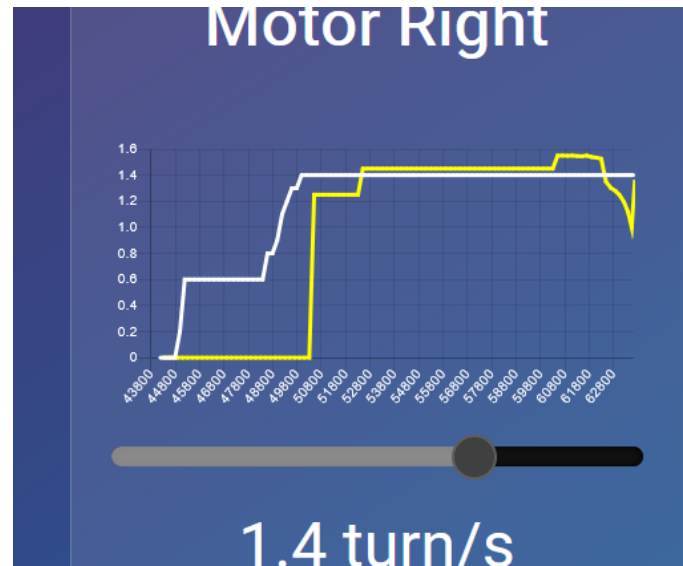


Figure 20 – Capture interface web des courbes permettant de déterminer les coefficients

La courbe monte rapidement puis se stabilise c'est pourquoi de manière empirique on peut dire que nous avons trouvé les bons coefficients KP, KI et KD qui sont respectivement dans l'ordre 0.1; 1.0; 0.1.

La fin du graphique n'est pas à prendre en compte, cet effet est dû à la mise à jour du graphique en permanence à une vitesse trop élevée et a été réglé depuis.

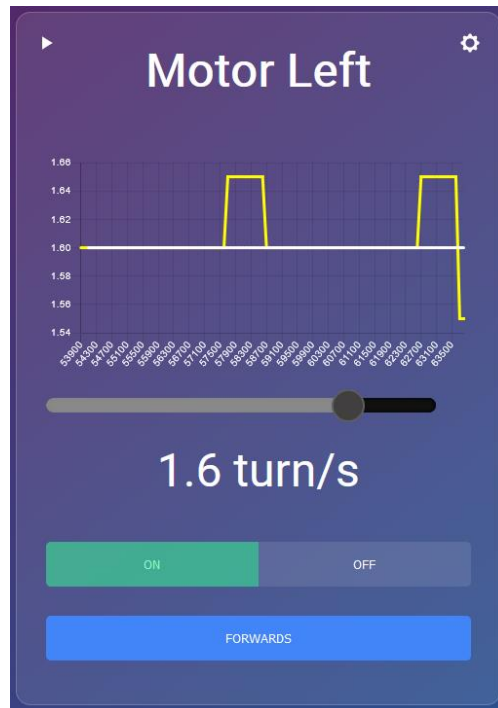


Figure 20 – Capture interface web illustrant la vitesse du robot

Cette capture d'écran permet de montrer que la vitesse du robot est plutôt stable malgré quelque variation dues à l'imprécision des capteurs.

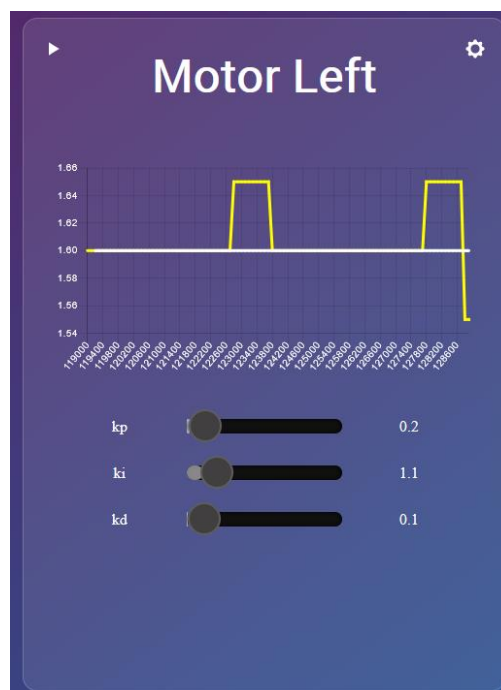


Figure 21 – Capture interface web illustrant le fonctionnement du traçage des courbes

Cette capture d'écran met en évidence le bon fonctionnement de la page web et du traçage des courbes de la vitesse et de la consigne.

C'est pourquoi on peut affirmer que la partie 3 est complétée.

VIII. Bilan

A. État d'avancement

Nous sommes aujourd'hui à la fin du projet voici un résumé de notre avancement. Nous sommes parvenus à compléter les trois premières parties du projet c'est à dire la mesure de la vitesse des moteurs ; l'interface Web ; Le PID pour chaque moteur. Malheureusement nous n'avons pas réussi à finir la partie quatre sur le Gyropode, nous avons élaboré notre code mais ce dernier ne fonctionne pas totalement cependant nous arrivons à quelques choses près à obtenir des valeurs. Nous pensons que cet échec est en relation avec la distribution tardive de "phone2" car nous avons pris du retard, nous aurions dû anticiper cela et continuer à avancer sur les autres modules. A l'avenir nous ne reproduirons pas cette erreur.

B. Pertinence de la solution technique

Ce projet a ses limites, en effet le matériel que nous utilisons ne nous permet pas de faire d'avantage d'innovations sur notre robot. Par exemple l'esp ne possède pas assez de pin pour rajouter plusieurs fonctionnalités.

C. Bilan sur le travail d'équipe

Ce projet a été très inspirant pour l'ensemble du groupe, tout le monde a su s'adapter, se comprendre. Nous tenons à souligner notre cohésion, lorsque l'un de nous rencontrait un problème difficile à résoudre l'ensemble du groupe se penchait sur le sujet dans le but de trouver la meilleure solution. Ce projet nous a permis de développer et d'approfondir un bon nombre d'aptitudes comme par exemple l'organisation ou encore la communication en groupe.

Tout ce que nous avons pu apprendre tout au long de ce projet nous servira à l'avenir que ce soit dans nos projets futurs avec l'école ou bien dans le monde professionnel. Ces qualités sont primordiales dans le monde du travail où tout le monde travaille dans un intérêt commun.

Néanmoins tout n'a pas été parfait. En effet, nous aurions pu mieux faire sur certains points comme par exemple le planning, nous avons profité des semaines sans

devoirs surveillés pour avancer sur le projet mais nous n'avons pas pu avancer comme nous le voulions, ce qui a engendré le fait que nous avons dû travailler sur une période de devoirs.

Pour un futur projet nous pensons que nous sommes capables de retravailler tous les quatre, la cohésion et la complicité unique que nous avons créé depuis le début de l'année nous aidera à réaliser les missions qui nous seront demandées. Cette fois l'organisation se fera mieux suivant les autres échéances.

IX. Sources

Documents utilisés et sites internet consultés pour développer le projet.

Brett Beauregard "Arduino Cloud IDE"

https://codebender.cc/library/PID_v1#PID_v1.cpp , 2016

"Asservir des moteurs à courant continu avec un PID et Arduino"

https://www.robot-maker.com/shop/blog/33_Asservir-moteurs-courant-continu-PID.html , 2017

"br3ttb/Arduino-PID-Library"

https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.cpp , 2017

« blanchon/ArduinoJson »

<https://github.com/bblanchon/ArduinoJson>

« jfturcot/SimpleTimer »

<https://github.com/jfturcot/SimpleTimer>

« me-no-dev/ESPAsyncWebServer »

<https://github.com/me-no-dev/ESPAsyncWebServer>

X. Annexes

Nous avons décidé de ne pas inclure notre code au sein même du rapport car ce dernier contient plus de 4000 lignes. Néanmoins nous mettrons dans les jours suivants notre dépôt GitHub en public dans lequel l'intégralité du code sera disponible à l'adresse suivante : <https://github.com/BragdonD/encoder-wheels> le Lundi 28/03/2022 à 18h00.