Documentation du Projet Hovercraft

BARBAUT Christophe
BAUDRI Romuald

3ème Année Option SMR

January 20, 2025

Abstract

Ce document présente les étapes de conception, les défis techniques et les résultats obtenus dans le cadre du projet Hovercraft. En utilisant une carte Arduino Nano 33 BLE Rev2, une centrale inertielle (IMU), ce projet vise à démontrer la faisabilité d'un suivi précis de position dans un contexte de navigation autonome. Des suggestions d'améliorations et des ressources pour poursuivre le projet sont également proposées.

Contents

List of Figures

1 Introduction (Christophe)

Il est important de rapeller qu'initialement le projet était prévue pour 4 personnes. Nous avons pris le risque de le faire à 2 et nous somme satisfait de nos résulats actuel.

1.1 Objectifs

- **Conception et réalisation d'un hovercraft autonome** :
  - Développement d'un système de sustentation efficace
  - Implémentation d'un système de propulsion contrôlable
  - Intégration d'un système de direction précis
- **Mise en place d'un système de contrôle intelligent** :
  - Développement d'une interface de contrôle sans fil via Bluetooth (BLE)
  - Implémentation de systèmes de sécurité pour les moteurs
  - Gestion des différents modes de fonctionnement (manuel/autonome)
- **Développement des capacités autonomes (très primitif)** :
  - Intégration d'un système de détection de ligne par capteur IR
  - Mise en place d'un compteur de tours pour le suivi de parcours
- **Optimisation des performances** :
  - Calibration précise des ESC pour un contrôle optimal
  - Ajustement des paramètres de sustentation pour une efficacité maximale
  - Développement d'un contrôle progressif pour la stabilité

1.2 Résumé des Étapes du Projet

On résume ici les différente étapes du projet, le matériel qui sera utilisé pour réponde au critères.

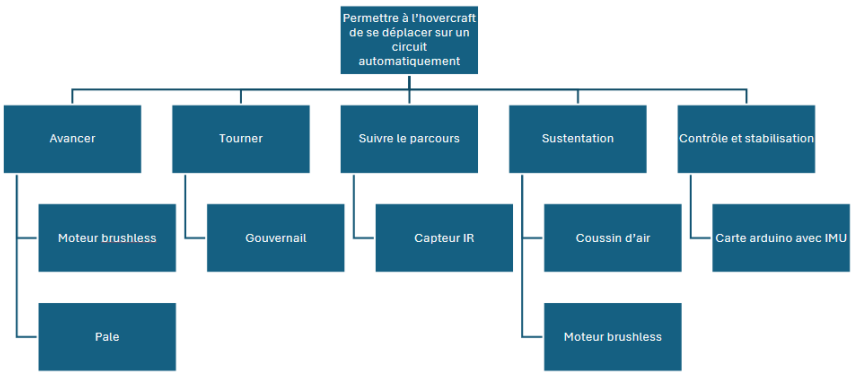

Figure 1: Diagramme Fast du projet

On peut résumé les étapes principale de ce projet dans ce diagramme PERT

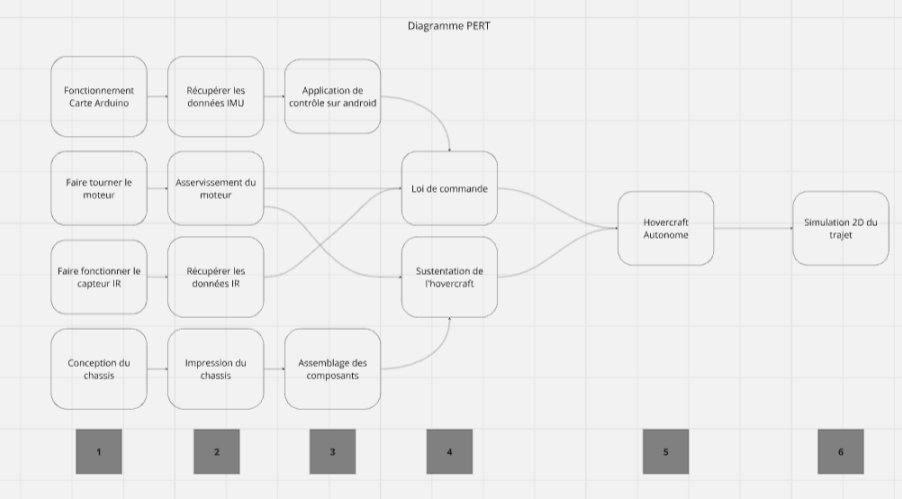

Figure 2: Diagramme PERT

1.3 Liste du Matériel

Nom	**Nombre**	**Tr**	**Description**	**Commentaire**
Moteur SmooX 1507 Plus - 3800KV	4	Validé	3800KV	Fonctionne avec une batterie 4S
GEMFAN 3052 - Tripales - Polycarbonate	2	Validé	Shaft M5, diamètre 3 pouces	x4 par unit
ESCs (Régulateur brushless)	4	Validé	Pichler XQ - Charge admissible (max.): 50 A Continue 40 A	
Servomoteur analogique	2	Validé	(L x l x H) 23 x 12 x 24 mm, Tension de fonctionnement 4.8 - 6 V	
Arduino Nano 33 BLE Rev2	2	Validé	Carte de développement basée sur un microcontrôleur Cortex-M4 avec Bluetooth 5 et IMU intégrée	Besoin d'un régulateur de tension pour l'alimentation

Table 1: Liste du matériel pour le projet Hovercraft.

1.4 CDCF

Fonctions	**Énoncé de la fonction**	**Critères d'appréciation**	**Niveaux d'exigence**
FP1	Se déplacer de manière autonome	Précision des déplacements	$<$ 5 cm d'erreur par rapport à la trajectoire cible
FP2	Éviter les obstacles	Capacité à détecter et contourner les obstacles	Obstacle détecté à $>$ 10 cm de distance
FP3	Détecter une ligne noir au sol	Capacité à suivre les lignes et à se relocaliser	Distinguer une bande de 2 cm de largeur à une distance de 5 cm
FC1	Être écologique	Consommation d'énergie réduite	Batterie rechargeable, durée de vie $>$ 1 heure
FC2	Coût de production abordable	Coût de fabrication par unité	$<$ 250€
FC3	Être esthétique	Design et finitions	Compact, design aérodynamique
FC4	Respecter les normes de sécurité	Sécurité des composants et utilisateurs	Composants non coupants, tensions $<$ 24V
FC5	Facile à programmer et contrôler	Temps nécessaire à la configuration	Moins de 30 minutes pour la configuration initiale
FC6	Résister à l'usure et à l'humidité	Robustesse des matériaux	Matériaux imperméables et résistants aux chocs
FC7	Avoir une documentation complète	Clarté des instructions et plans	Documentation technique détaillée et schémas CAO fournis
FC8	Installation faisable par des étudiants	Complexité des tâches	Niveau adapté pour des débutants en robotique et mécatronique
FC9	Avoir un poids réduit	Maniabilité et transportabilité	P1 : Poids total $<$ 750 g P2 : Poids total $<$ 500g
FC10	Loi de commande	Effectuer plusieurs simulation	Loi de commande correct

Table 2: Cahier des charges fonctionnel du projet

2 Développement

2.1 Conception Assistée par Ordinateur (CAO) (Romuald)

La conception des pièces de l'hovercraft à été un des plus gros challenges de ce projet. La conception des pièces de l'hovercraft a été l'un des plus grands défis de ce projet.

Après une discussion avec M.PIAT, le responsable du projet, pour définir les objectifs, il était clair que nous ne voulions pas créer un simple mini hovercraft. Aujourd'hui, on trouve de nombreux projets similaires sur internet, mais ils ont tous un point commun : ils sont radiocommandés et mesurent en moyenne 30 cm de long. Notre défi principal était donc de concevoir un hovercraft encore plus compact (20 cm de long), tout en intégrant l'ensemble du système Arduino pour le rendre automatisé. À notre connaissance, une telle réalisation n'a jamais été faite auparavant.

2.1.1 Première étape : l'analyse

Ma première démarche a été d'analyser ce qui existait déjà, car je n'avais aucune expérience préalable dans ce domaine. Voici quelques exemples d'hovercraft qui m'ont inspiré :

1er exemple : Mini hovercraft RC imprimé en 3D
Ce modèle respectait nos objectifs en termes de dimensions, mais il ne permettait pas de réaliser du sur-place ni d'accueillir tous nos composants. À partir de cette analyse, j'ai décidé d'opter pour un système utilisant deux moteurs : l'un pour la propulsion et l'autre pour la sustentation.

Figure 3: Miniature de la 1ère vidéo

2ème exemple : Hovercraft RC avec système de sustentation innovant
Ce projet était intéressant et respectait la plupart de nos exigences. C'est pour cela que j'ai commencé ma première conception en CAO en m'en inspirant. Cependant, je me suis vite rendu compte qu'il n'était pas adapté. En particulier, cet hovercraft n'était pas assez modulable. De manière générale, ce projet était trop brouillon et ne me permettait pas de bien intégrer toutes nos contraintes.

Figure 4: Miniature de la 2ème vidéo

3ème exemple : Hovercraft RC modulaire et Améliorations du modèle
C'est finalement sur la base de ce projet que j'ai conçu notre hovercraft. Ses principaux atouts étaient une CAO bien réalisée, un espace suffisant pour intégrer nos composants, et l'existence d'un forum actif autour de ce modèle, regroupant astuces, conseils et modifications. Le point

négatif était sa taille : il était beaucoup trop grand. J'ai donc dû recréer toutes les pièces à partir de zéro pour respecter nos contraintes.

Figure 5: Miniature de la 3ème vidéo

2.1.2 Première version de l'hovercraft

Une fois chaque pièce réalisée individuellement, ce qui m'a pris le plus de temps, j'ai adapté ma CAO pour rendre les pièces imprimables en 3D.

En effet, l'impression 3D a été choisie pour la fabrication des pièces en raison de ses nombreux atouts :

Avantages :

- Flexibilité de conception : Elle permet de réaliser des formes complexes difficiles à obtenir par d'autres procédés.
- Prototypage rapide : Les modifications peuvent être intégrées et testées rapidement.
- Coût réduit : Comparé à d'autres méthodes, elle reste économique pour des productions unitaires ou en petite série.

Inconvénients :

- Tolérances de fabrication : Les écarts dimensionnels liés à l'impression nécessitent des ajustements dans la conception (comme pour le diamètre des trous taraudés).
- Résistance mécanique limitée : Les pièces imprimées ne sont pas aussi robustes que celles fabriquées par injection ou usinage, ce qui limite leur utilisation dans des environnements exigeants.
- Temps de fabrication : Bien que rapide pour un prototype, l'impression 3D reste lente si plusieurs pièces doivent être produites.

À ce stade, la CAO contenait toutes les pièces principales. Cette première version avait pour objectif d'obtenir un aperçu global de l'hovercraft, afin de discuter plus facilement des premiers problèmes identifiés avec le reste de l'équipe, ainsi qu'avec notre aide technique, Pierre ROUX. Il était responsable de la conception mécanique et du prototypage rapide pour les différents projets PIST. Dans un second temps j'ai pu rentrer dans les détails de la conception et finir la première version.

2.1.3 Version 0.4

Dans cette version, les modifications suivantes ont été apportées par rapport à la V0.1 :

- Modification du diamètre de prt_upper_bati pour intégrer les tolérances de fabrication avec prt_lift_struct.
- Intégration du servomoteur dans la CAO.

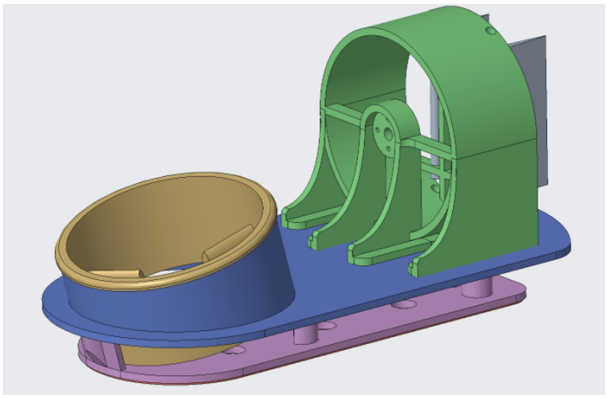

Figure 6: Version 0.1 de l'hovercraft

- Ajout de prt_rudders_link, une pièce permettant de lier les gouvernails au servomoteur.
- Création des trous pour la visserie.
- Ajout du capteur IR pour la détection de ligne.
- Ajustement des diamètres des trous taraudés (par exemple : diamètre 1,4 mm pour du M2 et 2,3 mm pour du M3).

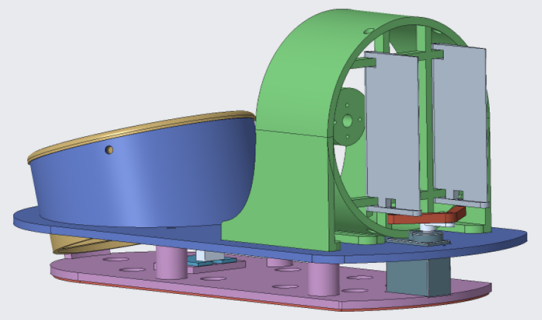

Figure 7: Version 0.4 de l'hovercraft, vue 1

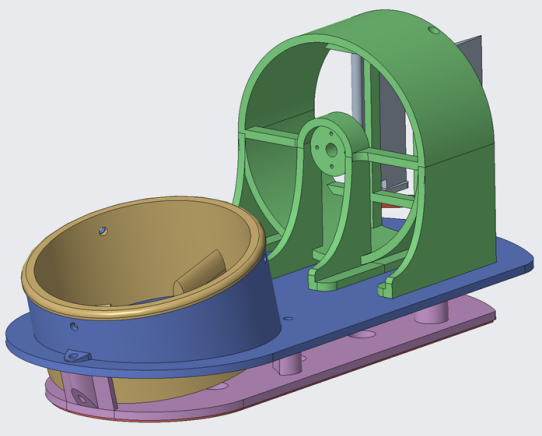

Figure 8: Version 0.4 de l'hovercraft, vue 1

2.1.4 Montage de l'hovercraft

Avec cette version 0.4 de l'hovercraft nous avons pu imprimer toutes les pièces à l'aide de Pierre ROUX, et commencer le montage du premier prototype. Le montage de l'hovercraft comprenait le câblage et la soudure des différents composants puis le vissage des différentes pièces entre elles. Avec enfin la fixation de la jupe à l'aide de clip.

Lors de cette étape, plusieurs problèmes ont été identifiés :

- **Défaut de fabrication de la pièce bleue :** Au niveau du flanc droit, la pièce était plus fine de 0,5 mm. Cela empêchait les clips de bien tenir la jupe en place. Avec M. ROUX, nous avons supposé que cette anomalie était due à une surchauffe de la pièce pendant l'impression 3D, ce qui a causé un bombement et une réduction de son épaisseur.
- **Fragilité des gouvernails :** En raison de leur forme particulière, M. ROUX a décidé de les imprimer en résine, ce qui les rendait très cassants. Lors du montage, nous avons cassé les deux pièces et avons même dû en rafistoler une avec de la colle.

Figure 9: Image du clip pour la fixation de la jupe

- **Difficulté de vissage :** Pendant la conception, j'ai parfois mélangé les trous taraudés et les dégagements pour les vis. Cela a entraîné des assemblages hyperstatiques, compliquant grandement le vissage.
- **Problèmes d'alignement des moteurs :** Les trous mal dimensionnés et mal positionnés ont causé des frottements entre les hélices et la structure. Pour corriger cela, nous avons dû poncer les parties gênantes.
- **Mauvaise intégration des ESCs :** En raison du surdimensionnement des ESCs, nous avons dû placer l'un des deux sur le dessus, aux côtés de la batterie, alors que nous avions initialement prévu de les positionner tous les deux en bas.

Voici le résultat final du prototype 1 :

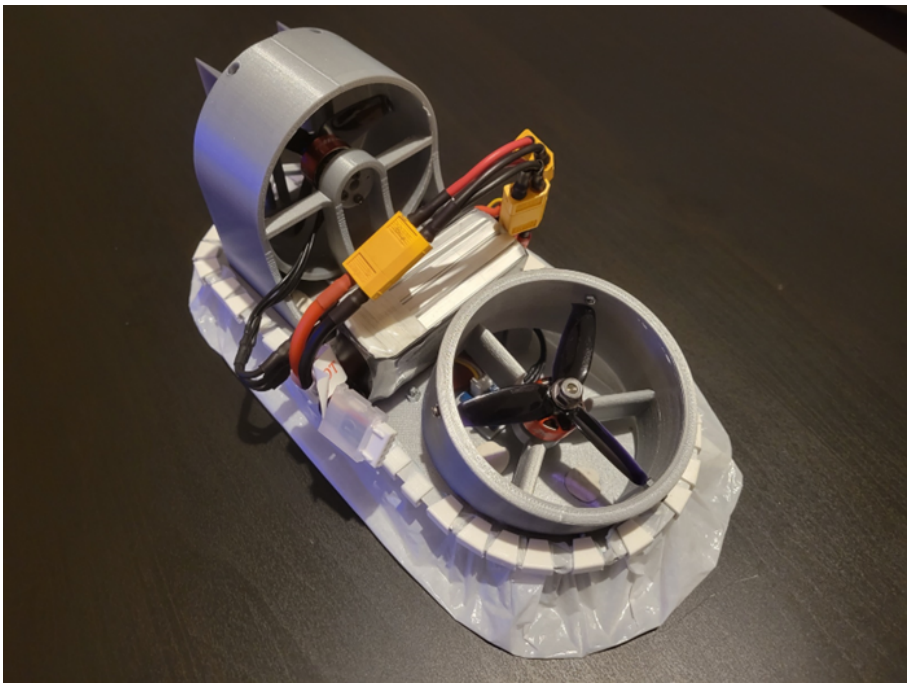

Figure 10: Résultat du prototypage

2.2 Code Arduino (Christophe)

L'arduino qui a été utilisé durant l'ensemble du projet est la Nano 33 BLE Rev2.

2.2.1 Première étape de mise en place du code

Le délai pour recevoir la bonne carte Arduino ayant pris du tant, nous avons commencer à travailler sur une Arduino Nano ESP32. L'objectif lors de cette étape était simplement de contrôler un servomoteur similaire à celui que nous utilisons à la fin.

Cela donne ce code qui effectue un contrôle basique d'un servomoteur en réalisant un balayage (sweep) entre deux angles définis, 45° et 135°. L'utilisation d'une interpolation permet un mouvement fluide entre les positions. Cette étape a permis de valider la compatibilité du matériel et de s'assurer que le moteur répondait correctement aux commandes.

Bien que ce servomoteur n'est pas limité en angles, ce qui on été acheté pour la suite du projet le sont. Il peuvent se déplacer de -45° à 45°.

```
#include <Servo.h>

Servo myServo;
int pos = 0;
int angleMin = 45;
int angleMax = 135;

// MAXIMUM ANGLE STEP FOR 25 DELAY IS 10° IN THE FOR LOOP (pos+=10) IN 1ms IT DOES 0.429° AT
↪  THE MAXIMUM

void setup() {
  myServo.attach(9);
  Serial.begin(115200);
  myServo.write((angleMin + angleMax) / 2); // Move to a neutral position
  delay(1000); // Let the servo stabilize
}

void loop() {
  static int currentPos = (angleMin + angleMax) / 2; // Start at neutral position

  // Sweep from angleMin to angleMax
  for (pos = angleMin; pos <= angleMax; pos += 10) {
    currentPos = currentPos + (pos - currentPos) / 2; // Smooth transition
    myServo.write(currentPos); // Write the smoothed position
    delay(25); // Stabilization delay
  }

  // Sweep back from angleMax to angleMin
  for (pos = angleMax; pos >= angleMin; pos -= 10) {
    currentPos = currentPos + (pos - currentPos) / 2; // Smooth transition
    myServo.write(currentPos); // Write the smoothed position
    delay(25); // Stabilization delay
  }
}

```

2.2.2 Utilisation du Bluetooth

L'un des premiers problèmes rencontrer était que l'arduino utilise du Blutooth qui est qualifier de BLE, autrement dit Bluetooth Low Energy.

Le Bluetooth Low Energy est une technologie de communication sans fil conçue pour minimiser la consommation d'énergie tout en permettant des échanges de données fiables sur de

courtes distances. Contrairement au Bluetooth classique, le BLE est optimisé pour des applications où les transferts de données sont intermittents ou nécessitent une faible bande passante, comme les appareils IoT ou les dispositifs portables.

Dans le cadre de ce projet, l'utilisation de l'Arduino Nano 33 BLE Rev2, équipée de cette technologie, a présenté plusieurs implications :

- **Configuration et Communication BLE** : Le BLE fonctionne autour du concept de services et de caractéristiques. Il a fallu concevoir une architecture logicielle spécifique où chaque commande de contrôle (comme la vitesse ou la direction) est associée à des caractéristiques BLE distinctes. Cela a nécessité la définition et l'implémentation de UUIDs (identifiants uniques) pour gérer les échanges de données.
- **Compatibilité des périphériques** : Tous les appareils ne supportent pas nativement le BLE, et des outils spécifiques, comme des applications mobiles ou des logiciels de débogage compatibles BLE. Cela a été le cas lors de la conception de l'application.
- **Limitations en débit de données** : Étant donné que le BLE est conçu pour des échanges légers, la transmission de données volumineuses ou continues, comme des flux de télémétrie en temps réel, n'est pas optimale. Cela a influencé la conception du système, privilégiant des commandes simples et efficaces pour le contrôle en temps réel.
- **Avantages pour l'hovercraft** : Grâce à sa faible consommation d'énergie, le BLE a contribué à allonger l'autonomie de la batterie de l'hovercraft, un avantage essentiel pour un dispositif mobile alimenté par une batterie.

Pour tester la compatibilité, nous avons effectué une première ébauche de code que vous pouvez trouver sur le dépôt GitHub du projet.

Ce code est une première ébauche visant à tester la communication Bluetooth Low Energy (BLE) entre l'Arduino Nano 33 BLE Rev2 et un périphérique externe. Il permet de :

- Contrôler un servomoteur en fonction des données reçues via le BLE, comme une position cible définie par une caractéristique BLE.
- Gérer des LED RGB via des caractéristiques BLE spécifiques pour simuler des actions et visualiser les données reçues.
- Assurer une transition fluide entre les positions cibles du servomoteur pour éviter les mouvements brusques.
- Établir une connexion BLE robuste en annonçant le service et en assurant la gestion de l'état connecté/déconnecté.

Ce prototype a permis de valider la communication BLE, d'établir un contrôle de base, et de poser les fondations pour une intégration plus complète des fonctionnalités du projet.

2.2.3 Contrôle du Moteur et des ESC

Le contrôle des moteurs brushless via les ESC a été une étape cruciale du projet. Voici un résumé des principales étapes effectués :

- **Choix des ESC** : Les ESC sélectionnés sont capables de supporter une charge continue de 40A avec un maximum de 50A, adapté aux moteurs SmooX 1507 Plus - 3800KV. Il a été vérifié que ces ESC disposent d'une compatibilité PWM pour les signaux de contrôle.

- **Signal de contrôle PWM** : Les moteurs brushless sont contrôlés via les ESC en utilisant un signal PWM. Ce signal, généralement compris entre 1000 µs et 2000 µs, correspond à la puissance sortant de l'ESC, dans ce cas 1000 µs correspond a une sortie de l'ESC de 0A et 2000 µs correspond a 40A. Un réglage fin des timings PWM a été effectué pour garantir une réponse fluide et précise.
- **Initialisation des ESC** : L'initialisation et la configuration des ESC sont des étapes cruciales pour garantir un fonctionnement optimal et sécurisé du système. Cette procédure se décompose en deux phases principales : la calibration initiale et la programmation des paramètres.

Phase de Calibration

La calibration est nécessaire pour que l'ESC puisse identifier correctement :

- La plage complète des signaux PWM (1000µs à 2000µs)
- Le point neutre (*idle*) à 1000µs
- La valeur maximale à 2000µs

Procédure de calibration : Il faut faire attention ici lors de la calibration, l'output de l'ESC en 5V ne doit pas être connecté, car l'ESC se met alros en tension depuis l'alimentation USB du PC et cela fausse la calibration.

1. Déconnexion de la batterie
2. Envoi du signal maximum (2000µs)
3. Connexion de la batterie
4. Attente d'environ 2 seconde maximum
5. Envoi du signal minimum (1000µs)
6. Confirmation finale par signal sonore

Phase de Programmation

Après la calibration initiale, l'ESC offre une interface de programmation accessible via une séquence spécifique :

- **Activation du mode programmation** :
  - Maintenir le signal neutre pendant 5 secondes après la mise sous tension
  - L'ESC émet des séquences sonores indiquant les différents paramètres
- **Paramètres programmables** :

Items \ Tones	"beep"	"beep,beep"	"beep,beep ,beep"	"beep,beep ,beep,beep"	"beep- -"	"beep- - beep"	"beep- - beep,beep"
	1short tone	2short tone	3short tone	4short tone	1long	1long 1short	1long 2short
SMR Function	***OFF**	ON					
Brake Type	***OFF**	Soft Brake	Mid Brake	Hard Brake			
Motor Timing	***Auto**	Low	Mid	High			
Motor Rotation	***CW**	CCW					
SR Function	ON	***OFF**					
Battery Cells	***Auto**	2S	3S	4S	5S	6S	
Low voltage Cutoff Threshold	OFF	NIMH50%	NIMH60%	***3.0V**	3.2V	3.4V	3.6V
Low Voltage Cutoff Type	***Reduce Power**	Cut off Power					
BEC Voltage	***5V**	6V					
Acceleration	***Normal**	Soft					
Restore Factory Default Sets	Restore						

Note: " * " value means default settings.

Figure 11: Tableau présentant l'ensemble des paramètres programmable

Points importants

- La calibration n'est nécessaire qu'une seule fois, sauf en cas de modification des paramètres
- Les paramètres sont conservés en mémoire même après déconnexion
- Une mauvaise calibration peut entraîner :
  * Des comportements instable du moteur
  * Une réponse non linéaire aux commandes
  * Des problèmes de démarrage
- La documentation complète de l'ESC est disponible en annexe pour référence

Note de sécurité : Il est crucial de toujours déconnecter les hélices lors de la calibration et de la programmation des ESC pour éviter tout démarrage accidentel des moteurs.

```
#include <Servo.h>

// Define signal ranges and pins
#define MAX_SIGNAL 2000
#define MIN_SIGNAL 1000
#define MOTOR_PIN 8
#define REVERSE_PIN 7  // Reverse signal wire connected to D7

Servo motor;

void setup() {
  Serial.begin(115200);
  delay(1500);

  Serial.println("Program begin...");
  delay(1000);
  Serial.println("This program will start the ESC.");

  motor.attach(MOTOR_PIN);

  // Configure reverse pin
  pinMode(REVERSE_PIN, OUTPUT);
  digitalWrite(REVERSE_PIN, LOW); // Start in forward mode by default

  // ESC calibration
  Serial.print("Now writing maximum output: (");
  Serial.print(MAX_SIGNAL);
  Serial.println(" us in this case)");
  Serial.println("Turn on power source, then wait 2 seconds and press any key.");
  motor.writeMicroseconds(MAX_SIGNAL);
  Serial.println("Step 1");

  // Wait for input to select the menu
while (!Serial.available());
  Serial.println("Step 2");
  Serial.read();
  motor.writeMicroseconds(MIN_SIGNAL);
  Serial.println("enter in the menu. Wait for selection of option");

  // Wait for input to select the option
while (!Serial.available());
    Serial.println("Step 3");
    Serial.read();
    motor.writeMicroseconds(MAX_SIGNAL);
```

```
45          // Wait for input to be back in the programm selection menu and then go back to
            ↪  calibration
46
47      while (!Serial.available());
48        Serial.println("Step 4");
49
50        Serial.read();
51        // Send minimum output
52        Serial.println("\n");
53        Serial.print("Sending minimum output: (");
54        Serial.print(MIN_SIGNAL);
55        Serial.println(" us in this case)");
56        motor.writeMicroseconds(MIN_SIGNAL);
57        Serial.println("The ESC is calibrated.");
58        Serial.println("----");
59        Serial.println("Type a value between 1000 and 2000 to control the motor.");
60        Serial.println("Send '1000' to stop the motor, '2000' for full throttle.");
61        Serial.println("Send 'R' to toggle reverse mode.");
62      }
63
64      void loop() {
65        if (Serial.available() > 0) {
66          String input = Serial.readStringUntil('\n'); // Read input as a string
67
68          // Handle reverse mode toggle
69          if (input.equalsIgnoreCase("R")) {
70            static bool isReverse = false; // Keep track of reverse state
71            isReverse = !isReverse; // Toggle reverse state
72            digitalWrite(REVERSE_PIN, isReverse ? HIGH : LOW); // Set reverse signal
73            Serial.print("Reverse mode ");
74            Serial.println(isReverse ? "activated." : "deactivated.");
75          }
76          else { // Handle throttle control
77            int throttle = input.toInt();
78            if (throttle >= MIN_SIGNAL && throttle <= MAX_SIGNAL) {
79              motor.writeMicroseconds(throttle); // Send throttle value to ESC
80
81              // Calculate motor speed percentage
82              float speedPercent = (throttle - MIN_SIGNAL) * 100.0 / (MAX_SIGNAL -
                ↪  MIN_SIGNAL);
83              Serial.print("Motor speed: ");
84              Serial.print(speedPercent);
85              Serial.println("%");
86            } else {
87              Serial.println("Invalid input! Enter a value between 1000 and 2000 or 'R' to
                ↪  toggle reverse.");
88            }
89          }
90        }
91      }
```

- **Problèmes rencontrés et solutions** :
  - **Problème** : Parfois, les ESC n'initialisaient pas correctement en raison d'une séquence incorrecte.
    Solution : Ajout d'une pause après la mise sous tension pour garantir une initialisation stable.
    Explication : Les ESC nécessitent une séquence d'initialisation précise pour calibrer leurs plages de fonctionnement. Cette séquence implique :
    * Envoi du signal maximum (2000µs)

    * Attente de la mise sous tension
    * Envoi du signal minimum (1000µs)
    * Attente de la confirmation sonore de l'ESC
  - **Problème** : Les moteurs avait tendence à vibrer lors d'une augmentation brusque de la puissance.
    Solution : Changer la valeur envoyée de manière plus incrémentale. Le moteur ayant du mal à passer de 0% à 20% d'un seul coup.
    Explication : Les moteurs brushless fonctionnent par commutation électronique. Un changement brusque de vitesse peut causer :
    * Une surcharge momentanée du système
    * Une désynchronisation entre le rotor et le timing de commutation
    * Des vibrations dues à la résonance mécanique
  - **Problème** : Surdimensionnement de l'ESC : Les ESC sont surdimensionnés pour le système. Ce qui pose un risque de brûler un moteur, ce qui à été le cas pour nous.
    Solution : Limitation logicielle de la puissance maximale via le code pour protéger les moteurs.
    Explication : Avec un ESC de 40A et un moteur dont la plage de fonctionnement optimale se situe entre 8-10A, avec une capacité maximale de 25A en pic. Pour garantir la durée de vie et les performances du moteur, il est recommandé de ne pas dépasser 12-15A en fonctionnement continu.
    * Un signal de 1500µs (50%) sur l'ESC peut délivrer jusqu'à 20A
    * Sous 14.8V (4S), cela représente une puissance de : $P = U \times I = 14.8V \times 20A = 296W$
    * La puissance maximale sécuritaire du moteur est de : $P_{max} = 14.8V \times 10A = 148W$
    * Solution implémentée : limitation du signal à 25% de la plage (1250µs max) pour maintenir le courant sous 10A
  - **Recommandations pour l'utilisation des ESC** :
    * Toujours commencer avec une valeur de throttle minimale
    * Modifier la commande pour appliquer un gradient éviter des variations trop brusques, réduisant ainsi les risques de dommage sur le moteur.
    * Surveiller la température du moteur et de l'ESC
    * Utiliser des connecteurs adaptés au courant maximal

Ces étapes ont permis d'obtenir un contrôle fiable et stable des moteurs pour assurer la propulsion de l'hovercraft.

2.2.4 Code finale du contrôle

Le code pour le contrôle finale de l'hovercraft peut-être trouvé ici.

Ce code permet maintenant le contrôle des deux joystick présent sur l'application. Un pour le contrôle des pâles et un pour la puissance délivrer au moteur.

Voici un résumé des différente fonctionnalité présente dans ce code.

Description du code Arduino : Le programme implémente le contrôle d'un hovercraft via Bluetooth Low Energy (BLE) avec les fonctionnalités suivantes :

- **Communication** : Utilisation du BLE pour établir une liaison sans fil avec l'appareil de contrôle

- **Contrôles principaux** :
  - Deux moteurs brushless contrôlés par ESC (Electronic Speed Controller)
  - Un servomoteur pour la direction
  - Un système de LED RGB pour le retour visuel, s'assurer que la carte est bien alimenté.
- **Fonctionnalités implémentées** :
  - Contrôle de la sustentation (moteur de gonflage). Cette partie de code pousse le moteur à fonctionner a 20%. pour cela on incrémente avec un léger délai la puissance envoyer afin de limiter les problème.
  - Contrôle de la propulsion (moteur principal)
  - Direction via servomoteur
  - Détection de ligne par capteur IR, on détecte dans notre cas des lignes noirs.
  - Retour d'état via LED RGB
- **Sécurité** :
  - Limitation des signaux ESC entre 1000µs et 1200µs
  - Initialisation sécurisée des ESC au démarrage
  - Contrôle progressif des changements d'angle du servo

2.3 Application Mobile (Christophe)

Présentation de l'application, avec explications sur l'interface et les fonctionnalités. Pour développer l'application nous avons utilisé MIT App inventor 2 (AI2).

2.3.1 Première Version

Cette version initiale présentait des fonctionnalités basiques : L'application implémente une connexion Bluetooth Low Energy (BLE) à travers plusieurs étapes :

Gestion de la Connexion Bluetooth LE :

1. **Processus de scan** :
   - Déclenchement du scan via le bouton "ButtonScan"
   - Mise à jour de l'interface utilisateur pendant le scan
   - Affichage des appareils détectés dans une liste (ListBLE)
2. **Établissement de la connexion** :
   - Sélection d'un appareil dans la liste
   - Connexion via "ButtonConnect" à l'appareil sélectionné
   - Indication visuelle de l'état de connexion (vert pour connecté, rouge pour déconnecté)

Fonctionnalités implémentées :

- Scan et connexion BLE
- Contrôle directionnel via un joystick virtuel
- Transmission des coordonnées X et Y du joystick

Architecture du joystick :

- Implémentation via un élément "Ball" contraint dans une zone
- Récupération des coordonnées lors du déplacement ("Dragged")
- Transmission des données via BLE lors du relâchement ("TouchUp")
- Envoi des coordonnées au format arrondi pour optimiser la transmission

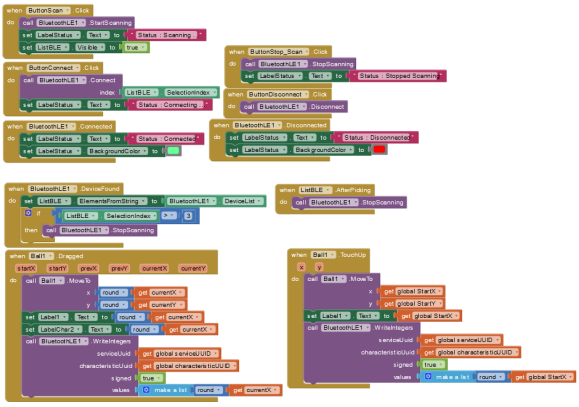

Figure 12: Code pour la première version de l'application

Limitations identifiées :

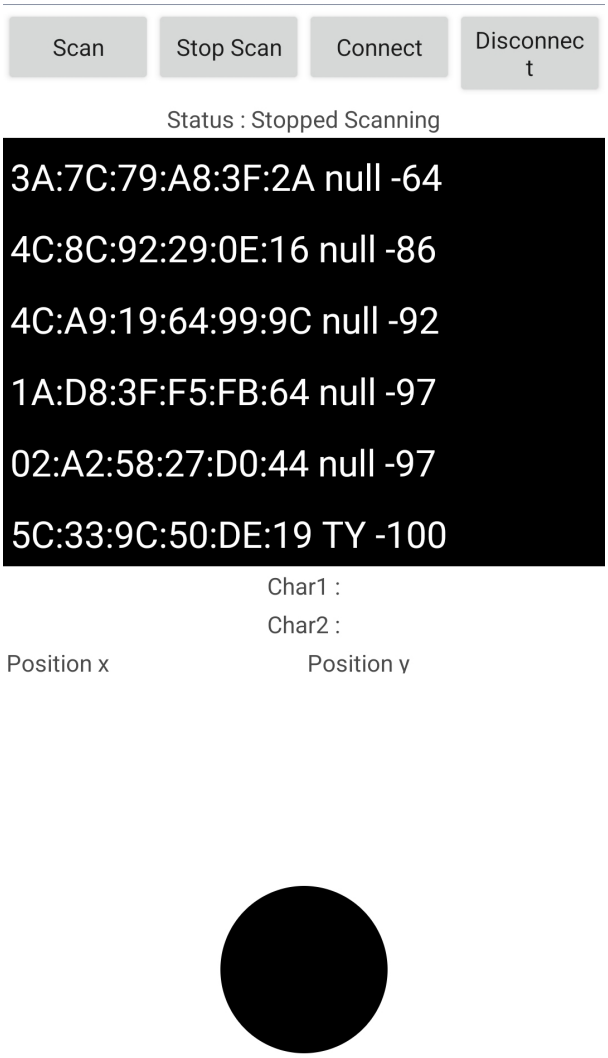

Figure 13: Première version de l'application

- **Problèmes d'interface** :
  - Scan trop rapide des appareils BLE
  - Processus de connexion peu ergonomique
  - Interface monolithique sans séparation des fonctionnalités
- **Limitations techniques** :
  - Mouvement non contraint du joystick
  - Manque de gestion des erreurs de connexion

Ces limitations ont guidé les améliorations apportées dans les versions ultérieures de l'application.

2.3.2 Deuxième version

Cette deuxième version permet une utilisation plus ergonomique pour la connexion. Différent onglets sont créer pour les différentes fonctionnalités.

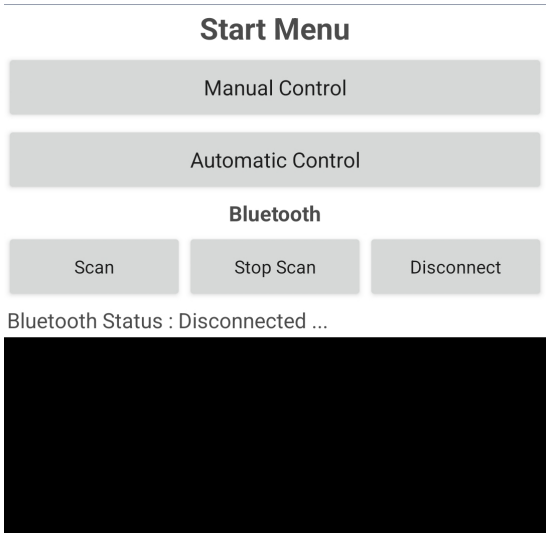

Figure 14: Start Menu

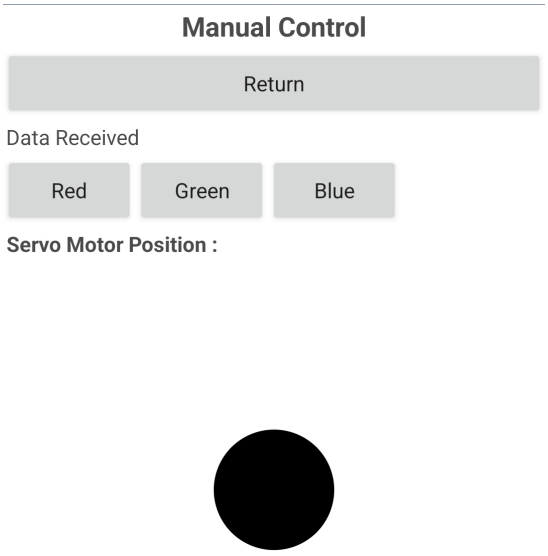

Figure 15: Contrôle Manuelle

Une difficulté majeure lors du développement a été l'utilisation de l'extension BLE (Bluetooth Low Energy) dans App Inventor 2 (AI2). Pour implémenter les onglets, une astuce a été employée : tous les onglets sont en réalité présents sur un seul écran. L'application utilise des sous-blocs qui peuvent être affichés ou masqués dynamiquement grâce à des boutons, simulant ainsi un système d'onglets.

Dans cette version, une amélioration notable est le mapping des valeurs du joystick, désormais réalisé directement dans l'application plutôt que sur l'Arduino. Ce mapping est essentiel pour convertir les données brutes du joystick en valeurs exploitables par le servomoteur, permettant un contrôle plus fluide et précis.

2.3.3 Version Finale

La version finale de l'application intègre plusieurs améliorations pour offrir une expérience utilisateur plus fluide et complète.

Dans cette version, une nouvelle extension d'App Inventor 2 (AI2) a été utilisée pour créer des joysticks plus ergonomiques. Chaque joystick permet un contrôle indépendant : l'un gère

le servomoteur, tandis que l'autre contrôle le moteur brushless responsable de la propulsion de l'hovercraft. Un bouton a été rajouter pour mettre en route l'inflation de la jupe de l'hovercraft.

De plus, une fonctionnalité dédiée au capteur infrarouge (IR) a été ajoutée pour le comptage de lignes. Contrairement aux versions précédentes, où l'application ne pouvait qu'envoyer des commandes à l'Arduino, cette version permet également au capteur de transmettre des informations vers l'application, qui les affiche en temps réel.

Enfin, une mesure de sécurité a été mise en place : l'utilisateur ne peut accéder aux contrôles de l'application que si la connexion Bluetooth est active, évitant ainsi tout comportement inattendu en l'absence de communication avec l'Arduino.

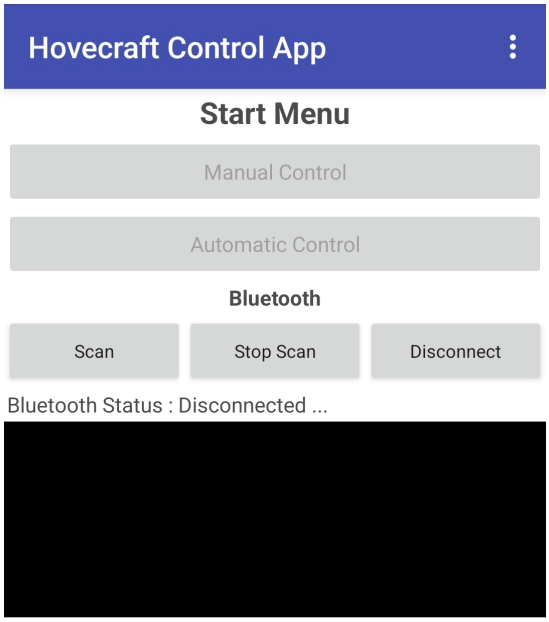

Figure 16: Start Menu 3rd Version

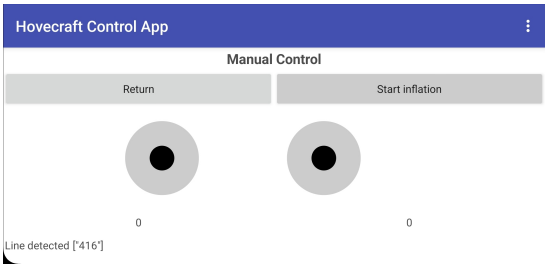

Figure 17: Manual 3rd Version

2.4 Câblage de la carte Arduino (Christophe/Romuald)

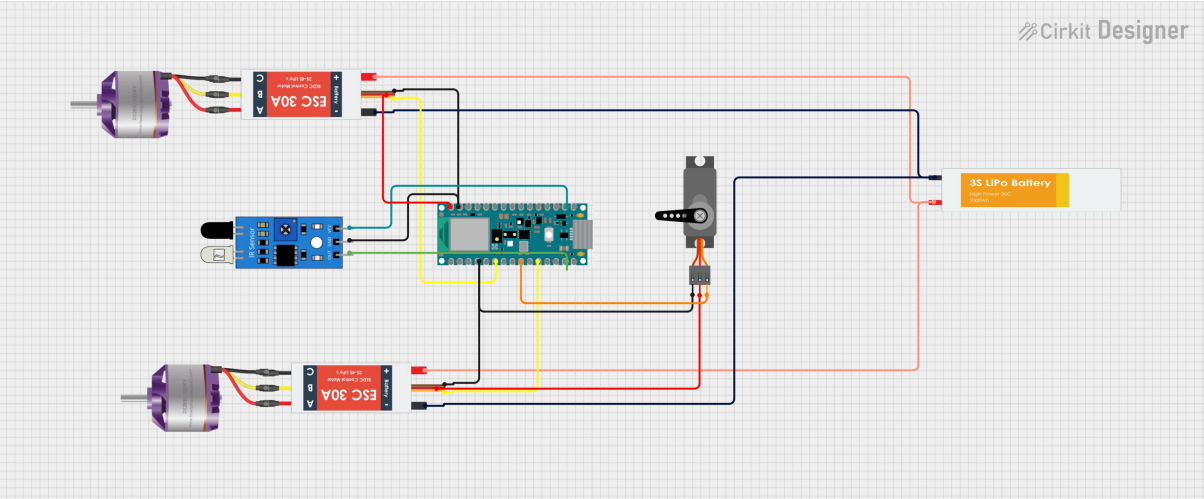

Figure 18: Image du circuit Sur l'arduino utilisé

Les câbles jaunes correspondent au signaux PWM. Le cable vert correspond au valeur reçu depuis l'IR. Les câbles rouges sortant des ESC sont des câbles qui alimente en 5V.

Pin	Type de Signal	Fonction
6	PWM (Jaune)	Contrôle du servo-moteur pour la direction (SERVO_PIN)
3	PWM (Jaune)	Contrôle du moteur ESC 1 pour la propulsion (MOTOR_PIN_1)
8	PWM (Jaune)	Contrôle du moteur ESC 2 pour le gonflage (MOTOR_PIN_2)
11	Digital Input	Capteur IR pour la détection de ligne (signalPin)

Table 3: Attribution des pins sur l'Arduino Nano

3 Axes d'Améliorations (Christophe/Romuald)

Suite au développement et aux tests du projet Hovercraft, plusieurs axes d'amélioration ont été identifiés pour optimiser les performances et la fiabilité du système. Ces axes ne sont que des pistes que nous n'avons pas poussé mais qui se basent sur notre expérience actuel et qui pourrait aider à guider la suite du projet.

3.1 Améliorations Mécaniques

3.1.1 Optimisation de la Sustentation

- Redimensionnement de la jupe pour une meilleure répartition de la pression d'air
- Utilisation de matériaux plus légers pour réduire la masse totale du système

3.1.2 Système de Propulsion

- Installation d'un système de refroidissement pour les ESC et moteurs brushless
- Optimisation de la position des hélices pour maximiser la poussée, les hélices actuels ne sont pas adaptées pour maximiser la poussé de l'hovercraft. Il faudrait prendre des hélices symétriques.
- Développement d'un système de protection des hélices contre les débris

3.2 Améliorations Électroniques

3.2.1 Gestion de l'Alimentation

- Implémentation d'un système de monitoring de la batterie en temps réel
- Ajout d'une protection contre les surintensités pour les moteurs
- Ajout de switch pour couper la batterie

3.2.2 Capteurs et Navigation

- Intégration de capteurs de proximité supplémentaires pour une meilleure détection d'obstacles
- Capacité à l'hovercraft de d'avancer et de reculer.
- Ajout d'un système de localisation par fusion de données (IMU + capteurs externes)

3.3 Améliorations Logicielles

3.3.1 Application Mobile

- Implémentation d'une interface de calibration des moteurs et servomoteurs
- Ajout d'un mode de diagnostic pour le dépannage
- Développement d'une fonctionnalité d'enregistrement des données de navigation

3.3.2 Firmware Arduino

- Implémentation d'un système de gestion des erreurs plus robuste
- Développement d'un mode de navigation autonome

3.4 Améliorations des Performances

3.4.1 Navigation Autonome

- Amélioration de la précision du comptage de tours

3.4.2 Contrôle et Stabilité

- Mise en place d'un PID pour le contrôle de la direction
- Optimisation des paramètres de contrôle pour une meilleure réactivité
- Développement d'un système de stabilisation automatique

3.5 Maintenance

3.5.1 Facilité de Maintenance

- Conception modulaire pour un remplacement facile des composants
- Amélioration de l'accessibilité aux composants critiques (Carte Arduino en particulier)

4 Conclusion

4.1 Rappel des objectifs et innovation

Le projet Hovercraft avait pour ambition de concevoir un système compact, autonome, et performant, capable de répondre à des critères exigeants d'innovation mécanique, électronique et logicielle. L'objectif principal ,pour nous, était de réaliser l'étude préliminaire et de créer une preuve de concept en réalisant un hovercraft contrôler manuellement.

Malgré des défis importants, notamment liés aux délais d'approvisionnement et aux contraintes de conception, nous avons réussi à créer un prototype fonctionnel et à poser les bases pour des évolutions futures.

4.2 Rétrospective sur le CDCF

Le Cahier des Charges Fonctionnel (CDCF) a guidé l'ensemble des phases de développement, servant de référence pour évaluer les progrès et les priorités. Parmi les objectifs principaux :

- **Réussites** :
  - L'hovercraft est capable de se déplacer.
  - La détection de lignes noires au sol a été implémentée avec succès, répondant aux critères définis.
  - Le contrôle à distance via une application mobile BLE est fonctionnel, offrant une interface intuitive et ergonomique.
- **Limites** :
  - Le poids total du prototype dépasse légèrement l'objectif de 500 g, en raison de la nature des composants utilisés et des contraintes du prototypage rapide.
  - La robustesse des matériaux, bien que suffisante pour des tests en environnement contrôlé, pourrait être améliorée pour des usages prolongés. En particulier les hélices guider par le servomoteurs

4.3 Retour sur le développement

Le développement de ce projet a permis de relever des défis techniques variés dans les domaines mécanique, électronique et logiciel :

- **Aspect mécanique** : La conception en CAO a permis de réaliser un châssis optimisé, bien que certains ajustements aient été nécessaires pour compenser les limitations de l'impression 3D.
- **Aspect électronique** : La calibration et le contrôle des ESC ont été maîtrisés (non sans mal), et les capteurs infrarouges ont été intégrés avec succès pour la détection de lignes.
- **Aspect logiciel** : L'application mobile BLE, couplée à des algorithmes de contrôle progressif des moteurs, a offert une interface stable et des performances satisfaisantes. Cependant, la gestion des interférences BLE pourrait être améliorée.

4.4 Impact du projet et perspectives

Ce projet a permis de développer des compétences techniques variées, notamment en conception mécanique, en programmation et en électronique embarquée. Il a également renforcé notre capacité à travailler en équipe et à résoudre des problèmes complexes. L'hovercraft ainsi conçu constitue une base solide pour des développements futurs.

4.5 Conclusion générale

En conclusion, ce projet a atteint ses objectifs principaux tout en offrant une expérience enrichissante pour l'ensemble des participants. Malgré certaines limites, les résultats obtenus témoignent de la faisabilité d'un hovercraft compact, intégrant des technologies avancées. Nous espérons que ce travail servira de point de départ pour d'autres équipes qui souhaiteront explorer et améliorer cette plateforme.

Annexes

A Bibliographie/Netographie

Différente vidéo qui ont inspiré le projet et permis de mieux comprendres les attentes

- Première vidéo How I 3D Printed a FAST RC Hovercraft .
- Deuxième vidéo 3D Printed RC Hovercraft on SPEED! .
- Lien vers le forum.

B Lien Github du Projet

Attention on différente branche du projet. La majorité se trouvant des les branches DEV-CAO et DEV-CODE

- Github du projet.

C Documentations Associées

Liens vers la documentation des composants utilisés. Certaines des documentations sont en Allemands. La majorité en Anglais.

- Drive avec les documentations.