Bouées géopositionnées pour régates de planche à voile

Lycée St Paul Besançon
Classe de 1ère spé NSI

Retrouvez-nous en vidéo sur : https://youtu.be/RfMReVmR9sE

CONCOURS 2026 Lycée CGÉNIAL

FONDATION CGENIAL

Sciences à l'École

Nos partenaires

LIGUE FFVoile Bourgogne Franche-Comté

Saint-Paul LYCÉE GÉNÉRAL ET TECHNOLOGIQUE

Problématique et contexte de la création

Problématique du projet :

Peut-on créer, avec des moyens "amateurs", un ensemble de bouées motorisées, délimitant des parcours de régates, qui maintiennent la position qui leur a été attribué et qui transmettent des données météo ?

Contexte scientifique du projet :

En Bourgogne Franche-Comté, plusieurs clubs de voile organisent des régates sur des lacs. Cependant, ancrer des bouées pour délimiter les parcours soulève la vase, gène la reproduction des poissons, prend du temps et nécessite du personnel. Asservir la position de chaque bouée à une coordonnée GPS sans les ancrer élimine ces problèmes. Celles-ci pourraient aussi capturer et transmettre des informations météo vers le comité de course afin d'augmenter la sécurité de la compétition. Ce type de bouée existe (sans la prise de mesures) mais elles sont très chères

Caractère innovant du projet :

Ce projet permettrait à des clubs de s'équiper à moindre coût d'un matériel écologique permettant de construire rapidement un parcours de régate. L'innovation consiste surtout dans l'utilisation de matériels du commerce et de logiciels libres que nous devrons faire fonctionner ensemble. Ce projet s'inscrit donc dans la création d'objets dits "intelligents".

Genèse du projet

Lors d'une assemblée de la ligue des bouées géopositionnées ont été présentées aux clubs de Bourgogne-franche-Comté (BFC) dans le but d'un éventuel achat. Ces bouées d'occasion valaient 4000€ pièce. Il en faut un minimum de cinq pour créer un parcours simple.

Aucun club n'a pu faire cet achat faute de budget.

L'idée d'une création amateure flottait dans l'air. Lors d'une discussion a la fin d'un week-end de régate, des clubs ont lancé le défit d'en fabriquer pour moins de 600€ pièce. Nous l'avons relevé.

En effet, les clubs de la région BFC naviguent sur des lacs ou des rivières. Ils n'ont pas besoin d'une bouée aussi puissante et performante que ceux qui naviguent en mer. Il y a donc un « marché » pour des bouées moins performantes mais moins chères.

De plus nous pourrions proposer des bouées faisant des mesures météo ce qui n'existe pas.

L'idée serait de faire un DIY afin que chacun réalise ce dont il a besoin. Cependant des contacts ont été pris avec une entreprise bisontine pour ceux qui ne pourraient pas réaliser les parties électrique et électronique en interne.

Cahier des charges :

Créer une bouée positionnée par GPS permettant de délimiter des parcours de régate (on se limitera à la réalisation d'un DIY (do it yourself : procédure permettant à chaque club de construire lui-même ses bouées) et d'un prototype.

Cette bouée doit pouvoir maintenir sa position ou y revenir rapidement, même si le vent, le courant ou un bateau la déplace.

Elle doit pouvoir se rendre seule à un emplacement désigné sur une carte mais doit aussi pouvoir être remorquée , si le trajet est trop long, afin d'économiser sa batterie.

Elle n'est pas ancrée au fond afin de ne pas soulever la vase ni endommager les herbiers.

Elle est alimentée sur batterie et éventuellement par panneau solaire. Elle est propulsée par des moteurs électriques.

En option, elle peut transmettre des informations au comité de course, comme la force et la direction du vent qu'elle reçoit, l'état de sa batterie, etc… elle peut relayer les transmissions d'autres bouées (fonctionnement en réseau).

Voici l'idée de départ --->

Le choix du matériel :

On utilise un microcontrôleur raspberry-pi_3B+ car il est peu cher et se programme en python (c'est le seul langage que nous connaissons). On en a plusieurs en stock au lycée.
La structure sera en bois enduit de résine et en chambres à air.

La faisabilité a été validée par notre professeur : le projet sera long et difficile, cela va être tendu mais on peut y arriver. Au pire on pourra supprimer des options. Le coût financier est acceptable.

Nous aurons le droit d'utiliser une IA pour nous aider à coder, surtout pour les parties « robotique » qui ne sont pas au programme de 1ère NSI (Utilisation du GPIO*, utilisation des bus SPI, I2C* et UART*, des modules LoRa* et GPS. Elle doit se limiter à une aide.

Nous allons écrire les codes Python, nécessaires pour chaque élément, on va les tester (tests unitaires) puis on les réunira pour produire les codes complets de la bouée et de la console (test d'intégration).

*
- le GPIO est le bornier d'entrée/sortie du raspberry. Il comporte 40 broches .
- LoRA est l'abréviation de long Range. C'est un système de communication entre deux modules par ondes radio. Sa portée est de plus d'un kilomètre.
- Les bus SPI et I2C sont des moyens de communication synchrone entre le microcontrôleur et ses périphériques.
- Le bus UART est un système de communication asynchrone entre le microcontrôleur et un seul périphérique. Il est plus simple que les précédents mais ne nous suffit pas.

Présentation de l'équipe

- **ANDRE Louis** Membre de l'équipe console a beaucoup transpiré sur les problèmes de transmission
- **CHIEUX Milo** Membre de l'équipe bouée Mise au point des capteurs et des mesures (anémomètre, boussole…)
- **GODDE Eloïse** Membre de l'équipe console, elle a aussi réalisé la vidéo
- **JEANNIN Joseph** Membre de l'équipe console. Mise au point de l'affichage. Co-rédacteur de ce document
- **MAGNERON Victor** Membre de l'équipe bouée a beaucoup transpiré sur les problèmes de transmission
- **NICOD Vincent** Membre de l'équipe bouée, Mise au point des capteurs et des mesures (anémomètre, boussole…), co-rédacteur de ce document
- **M. LE BRIS** Professeur de NSI, il est aussi arbitre régional de voile.

Chronologie

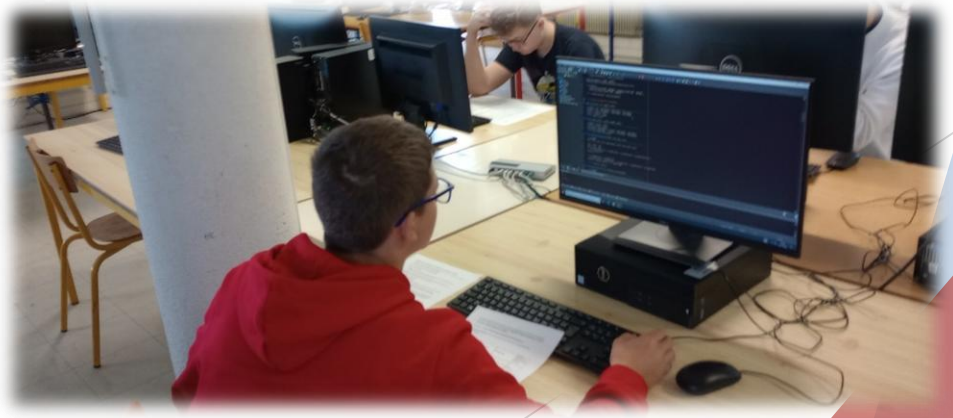
Chronologie :

14 novembre : étude de la console : comment afficher une image sur l'écran lié au raspberry. Comment convertir les coordonnées GPS associées à l'image en coordonnées pixel associées à l'écran. On a écrit l'algorithme général et on se fait aider par une IA (Grok) pour l'implémenter . Bug monumental, l'image est retournée verticalement .

En effet, en Python le zéro est en haut à gauche et l'axe des ordonnées descend... facile à corriger mais il fallait y penser. Grock n'a pas trouvé le problème, il a fallu le lui expliquer. On a décidé de l'abandonner car on allait plus vite sans elle.

02 décembre : initiation à la programmation orientée objet : On voit rapidement les bases, c'est normalement du programme de terminale donc on se limite au strict nécessaire.

05 décembre : mise en œuvre du GPIO . Nous avons fait plusieurs exercices (lire un niveau logique sur une patte, faire clignoter une led…) On a aussi grillé un GPIO. Morale de la séance : ne jamais brancher quoi que ce soit avec le raspberry sous tension, le GPIO n'a aucune protection…



12 DÉCEMBRE : Le week end dernier, notre professeur a présenté le projet à la journée de clubs de voile, en présence du président de la ligue de bourgogne-franche-Comté. Le retour a été très positif bien que l'aspect « bibendum» (c'est leur expression) du projet les aient fait rire. La ligue décide de nous soutenir. Il faudra qu'on corrige l'allure de la bouée.

Le club « impression 3D » a terminé la fabrication de la girouette et de l'anémomètre. Le groupe bouée crée les programmes nécessaires.

Pour la girouette des leds éclairent des capteurs, des trous dans une portion de cercle portée par la girouette laissent passer cette lumière ou pas. Cela forme un code binaire. Il suffit de lire ce code sur 4 pattes du GPIO

Pour l'anémomètre, un aimant porté par sa structure tourne devant un interrupteur ILS. Cela crée des impulsions de tension qu'il suffit de compter. On test un programme de lecture des données de ces deux appareils que nous avons écrits en adaptant des exemples trouvés sur internet. Cela fonctionne mal, on n'arrive pas à interpréter les résultats qui semblent aléatoires. De plus on a cassé trois ILS au montage (petits tubes de verre, très fragiles).

16 DÉCEMBRE : Aide de l'IA « géminy » pour tenter de résoudre le problème de l'anémomètre : rien d'efficace.

La solution vient du site « framboise.pi, le raspberry à la française »: sur les montages électroniques de la girouette et de l'anémomètre, notre professeur a placé des résistances de pull-up afin de garantir les niveaux logiques. Cependant il y a des résistances internes au GPIO qui font le même travail mais dans l'autre sens (pull-down) et on les a activées dans notre code (parce que c'était conseillé dans les exemples qu'on a trouvé sur internet). Ce n'est pas compatible. On a retiré les résistances sur la carte comme conseillé par « framboise.pi » et cela fonctionne (mais on a aussi remplacé la lecture magnétique de l'anémomètre par une lecture optique (bande blanche passant devant un photo-détecteur) car c'est plus léger et plus solide.

Maintenant c'est plus fiable mais, de temps en temps, l'anémomètre loupe un front montant ce qui fausse légèrement la mesure).

L'affichage de la photo avec les bouées en surimpression fonctionne (groupe console).

09, 16 et 23 janvier : un groupe travaille sur le GPS, l'autre sur les cartes LoRa et la transmission entre deux raspberry. Beaucoup d'aide de l'IA (Gémini.

Le GPS fonctionne assez rapidement à partir du moment ou nous avons compris qu'il fallait aller dans le couloir (plafond plus perméable aux ondes.

On a du mal à faire fonctionner le LoRa car il faut modifier les fichiers de configuration du raspberry (commandes bash en mode sudo) et il faut penser à bien configurer les cartes. D'une séance à l'autre Gemini oublie les consignes même si on reprend la conversation. Il faut tout lui réexpliquer. Il a tendance à oublier des points du cahier des charges et de l'algorithme quand on lui demande de simplifier son code. L'IA n'est donc pas un codeur infaillible, loin de la. Cependant il explique bien.

Début janvier notre professeur à enfin pu avoir des chambres à air correspondant à nos besoins, il a pu donner les dimensions aux profs de BTS CPI qui doivent étudier la structure de la bouée.

Nous sommes en retard sur l'avancement prévu, notre professeur nous aide un peu en debuggant les codes entre les séances.

Dans quelques semaines, Les BTS nous proposeront ceci ---->
Au final on n'utilisera que des chambres a air.

30/01 et 06/02 : intégration des différentes composantes des codes afin de créer un code homogène pour la bouée et un autre pour la console. Pour la bouée on laisse l'IA coder la partie « moteurs » car ce n'est pas dans nos compétences ni à notre programme. Le GPS fonctionne, la console aussi…. Mais la console ne reçoit rien de la bouée. Notre prof nous fait remarquer que le groupe bouée a appelé ses bouées « BOUEE01, BOUEE02, etc » alors que le groupe console test si les messages qu'il reçoit commencent bien par « B01, B02, etc ». Donc il rejette les messages, vu qu'ils ne commencent pas par le bon indicatif. On décide d'adopter la notation en B0n, c'est plus simple. Cependant, le 30 à midi cela ne fonctionne pas, il doit y avoir une autre erreur. Notre professeur reprend cela entre les deux séances, et se rend compte que la console est tout le temps en mode émission donc elle ne peut pas recevoir. Il faut modifier le code : Passer en émission que si on appuie sur une touche sinon la console écoute.

27 février (semaine de la rentrée) : Les modules LoRa ne fonctionnent toujours pas avec la console mais fonctionnent seuls. étude de l'algorithme des programmes console et bouée afin de détecter des incompatibilités. Éloïse travaille sur la vidéo, en partie avec Milo et Vincent pour trouver les fautes d'orthographes

6 mars : On reste bloqué sur la recherche de la panne LoRa d'autant que maintenant plus rien ne fonctionne. On essaye beaucoup de modifications en demandant de l'aide un peu partout, il en ressort que nous avons un problème de bus. Discussion sur les bus et leur rôle. Par acquis de conscience on commande un nouveau module LoRa car on a un doute sur l'état du module de la bouée.

10 mars : la panne est trouvée Elle était double : La première panne était un problème de registre et non de bus. On écrivait le message à envoyer dans le registre RX (code 0x00 qui est réservé à la réception). Il aurait fallu écrire dans le registre TX (transmission) code 0x80. Une erreur d'inattention qui nous a couté pas mal de temps. La deuxième panne est liée à la première par le stress : à force de démonter et remonter le module LoRa de la bouée, on l'a endommagé. Il a fallu le changer. Cela fonctionne. On monte la boussole mais on n'a pas le temps de la tester.

13 mars : rédaction du rapport de projet, tests moteurs, cette fois c'est la boussole qui ne fonctionne pas, après enquête avec l'aide d'une IA qui balaye les forums, notre capteur vient d'un lot qui pose des problèmes avec le raspberry. On abandonne cette option car nous n'avons pas le temps d'en chercher un autre.

La version actuelle ---->

Rapport financier

designation	quantité	prix/u TTC	Total TTC	commentaire
connecteur XT60 lot de 20	1	17,99 €	17,99 €	lot de connecteurs pour relier les moteurs aux batteries et au chargeur
Chambre a air 18/8,50/8 lot de 2	1	13,99 €	13,99 €	chambres a air de tracteur-tondeuse pour le haut de la bouée
Chambre a air 195/45R14	2	20,32 €	40,64 €	Chambre a air de voiture : bas de la bouée
batteries lithiumFerPhosphate LiFePO4 30Ah Lot de 2	1	159,99 €	159,99 €	batteries de propulsion
Chargeur de batterie lithium	1	59,99 €	59,99 €	Chargeur de batteries avec sécurité anti incendie (Lithium)
Convertisseur MCP3008	2	6,47 €	12,94 €	Convertisseur analogique numérique pour mesurer la tension de la batterie ("état de la batterie")
Moteurs apisqueen U01 CW avec circuit ESC	1	55,98 €	55,98 €	moteur gauche de la bouée (tourne dans le sens des aiguilles d'une montre (CW))
Moteurs apisqueen U01 CCW avec circuit ESC	1	55,98 €	55,98 €	moteur gauche de la bouée (tourne dans le sens inverse des aiguilles d'une montre (CCW))
Boussole magnétometre 3 axes lot de 2	1	9,89 €	9,89 €	
Module LoRa lot de 2	1	19,31 €	19,31 €	communication entre la bouée et la console
Antennes LoRa lot de 2	1	10,99 €	10,99 €	
Module GPS NEO MM	1	10,99 €	10,99 €	capteur de position GPS
Interface I2C pour moniteur de puissance (lot de 3)	1	9,55 €	9,55 €	Pour pouvoir utiliser les modules MCP3008
Tissus jaune fluo étanche 1m*1,5m	1	14,00 €	14,00 €	
raspberry PI 4B+ kit complet	0	119,90 €	- €	issu des ressources du lycée
raspberry PI 3B+	0	59,90 €	- €	issu des ressources du lycée
carte mémoire pour raspberry 3B+	0	16,50 €	- €	issu des ressources du lycée
Contreplaqué pour la structure	0		- €	fournie par le lycée professionel saint Joseph
Divers cables et électronique	0		- €	issu des ressources du lycée ou du professeur
		Total	492,23 €	

recettes	
designation	montant
subvention concours C'Genial	300 €
Subvention lycée	200 €
Total	500 €

recettes	500 €
dépenses	492,23 €
solde	7,77 €

Plusieurs éléments de la bouée présentent un cout nul, car sont déjà dans les réserves du lycée. De plus le contreplaqué pour la structure de la bouée, qui nous a été fourni par le lycée professionnel, n'a pas engendré de frais supplémentaires.

Détail des parties

(l'ordre est un peu fictif car nous avons formé deux à trois groupes travaillant en parallèle)

1. Phase de concertation collective :

Définition du cahier des charges fonctionnel de la bouée, étude des capacités de la bouée, inventaire des matériaux…

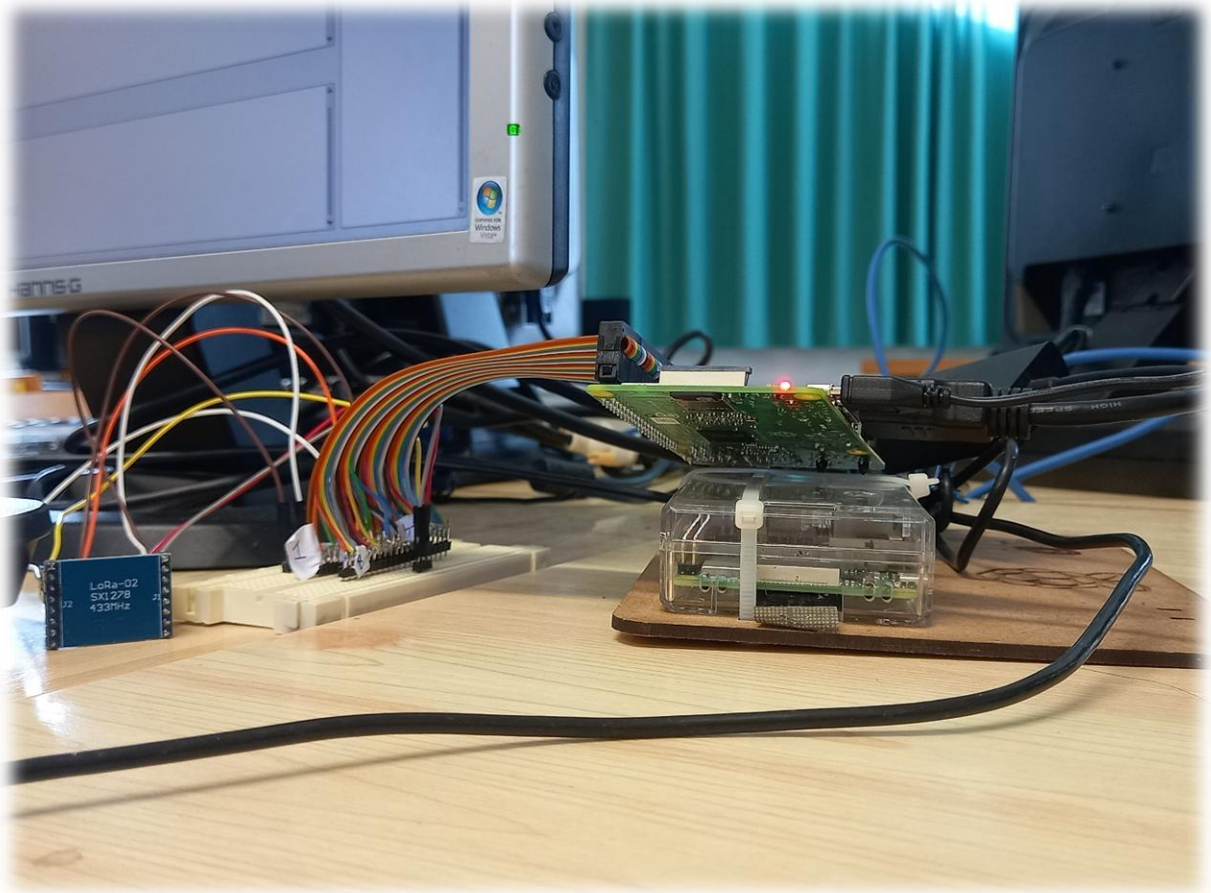
Méthode générale :

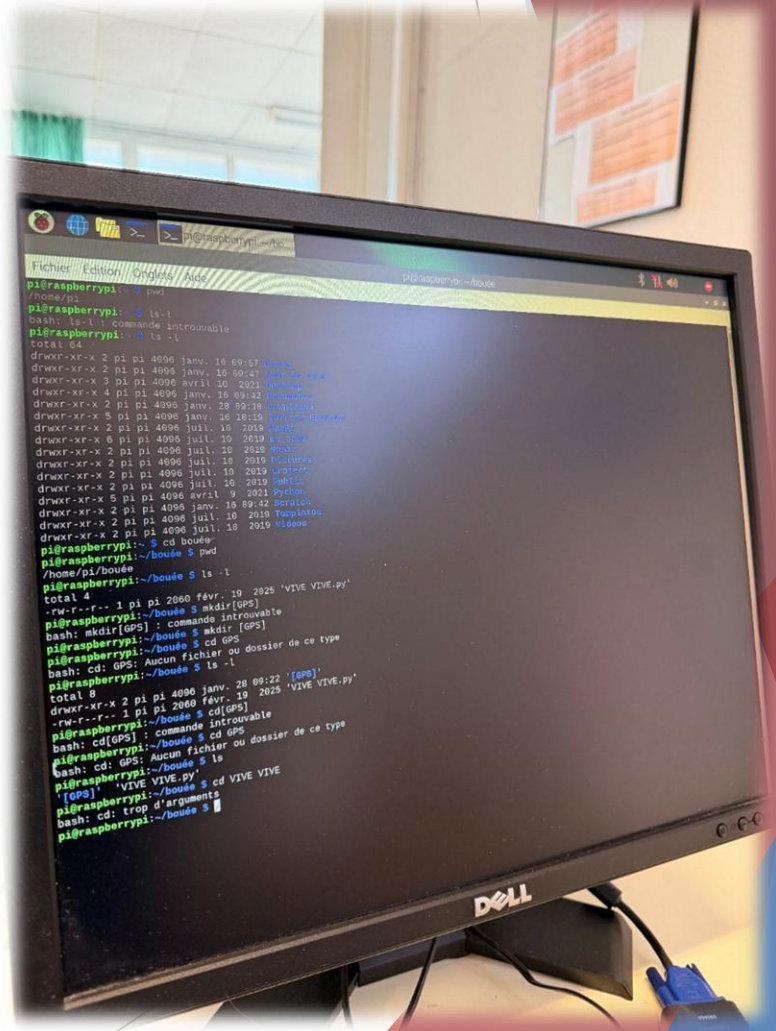
(Chaque partie est détaillée dans les pages suivantes)

- Nous avons déterminé les différentes parties de la bouée et de sa console.
- Cela nous a donné nos besoins matériels et logiciels
- Nous avons commandé le matériel
- Nous avons fait les algorithmes des programmes
- Nous les avons implémentés (transcrits en code informatique) en langage Python. A ce stade, nous avons demandé à une IA (Gémini) des cours ou des informations supplémentaires sur le matériel.
- Nous les avons testés (Gémini et Grok nous ont servi à créer les procédures de tests afin de ne rien oublier et nous ont aidé dans la chasse aux bugs notamment pour comparer rapidement plusieurs programmes)
- Plus tard il faudra réaliser les montages électroniques sur des supports permanents et les enfermer dans des boitiers étanches

2. Acquisition de compétences techniques :

Prise en main de l’écosystème Linux et gestion de l’interface entre les modules et capteurs

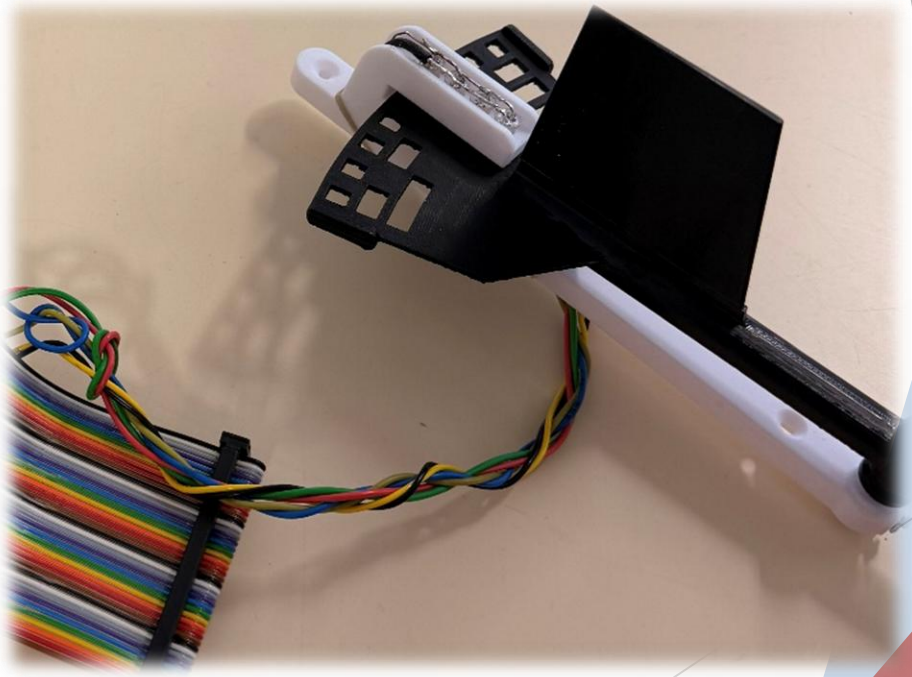




2. Conception de solutions opérationnelles répondant aux enjeux identifiés

Exemple : Afin d'éviter l'achat d'une girouette à direction complète(360°) précise à 2°, trop onéreuse, nous utilisons une girouette à direction limitée, pour orienter la bouée vers le vent.
Un module boussole (intégrée à la bouée permettra alors de connaître la direction de notre bouée et donc du vent.

En effet une girouette 360°, précise à 2°, coute environ 150€, un module boussole, précis à 2°, fonctionnant sur 360° coûte 12€ et notre bricolage à peu près 5€…



3. Approvisionnement du matériel nécessaire à la fabrication de la bouée

Système de propulsion, connectivité et structure

Capacité de la batterie	30AH
Puissance de la batterie	384WH
Tension de la batterie	12.8V
Courant de charge maximal	30A
Courant de décharge maximal	30A
Tension de charge maximale	14.6V
Tension de coupure de décharge	10V
Plage de température de charge	0-55°C
Plage de température de décharge	-20-55°C(-4°F-131°F)
Taille de la vis	M5
Dimensions de la batterie	7.1*3.0*6.5 inch(180*76*166 mm)
Poids de la batterie	6.61 pounds (3.0KG)
Taille du port CC	5.5*2.5mm
Courant de charge/décharge du port CC	5A

Bonne surprise, les batteries possèdent une prise annexe, nous ne seront pas obligés d'acheter une autre batterie pour l'électronique.

4. Phase de conception logicielle assistée par l'IA « Gemini » : résolution de bugs, enrichissement fonctionnel et validation par comptes-rendus de tests.

```python
# console_V10.1 - 06/02/26
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from matplotlib.gridspec import GridSpec
import numpy as np
import time, spidev, math, random
import RPi.GPIO as GPIO

# ========================================
# CONFIGURATION
SIMULATION_MODE = True
# ========================================

class ConsoleV10:
    def __init__(self):
        print("--- CONFIGURATION DU TERRAIN ---")
        self.lat_min = float(input("Coin BAS-GAUCHE - Lat : "))
        self.lon_min = float(input("Coin BAS-GAUCHE - Lon : "))
        self.lat_max = float(input("Coin HAUT-DROITE - Lat : "))
        self.lon_max = float(input("Coin HAUT-DROITE - Lon : "))
        self.map_extent = [self.lon_min, self.lon_max, self.lat_min, self.lat_max]

        self.center_lat = (self.lat_min + self.lat_max) / 2
        self.center_lon = (self.lon_min + self.lon_max) / 2

        self.nb_bouees = int(input("\nNombre de bouées (en plus de B00) : "))
        self.distinct_finish = input("Bouée d'arrivée distincte ? (o/n) : ").lower() == 'o'

        # Initialisation du dictionnaire
        self.bouees = {f"B{i:02d}": {
            "lat": self.center_lat, "lon": self.center_lon,
            "status": "En attente", "batt": 100, "vent_vit": 0.0,
            "cap": 0.0, "active": False, "last_seen": 0, "mode_test": False
        } for i in range(self.nb_bouees + 1)}

        self.setup_ui()

    def setup_ui(self):
        plt.rcParams['toolbar'] = 'None'
        self.fig = plt.figure(figsize=(16, 9))
        gs = GridSpec(4, 6, figure=self.fig)
        self.ax_dist = self.fig.add_subplot(gs[0:3, 0])
        self.ax_map = self.fig.add_subplot(gs[0:3, 1:6])
        self.ax_table = self.fig.add_subplot(gs[3, :])
        try:
            self.img = mpimg.imread('Lycee.png')
        except:
            self.img = None
```

Voici, à droite, un programme où l'on voit les spécifications de la version 5 du programme de la bouée. À gauche, on trouve la version 10.1 du programme de la console. On observe les fonctions et les classes utilisées pour son fonctionnement.

```python
# bouee_V5.py
# Ce code intègre  Navigation simplifiée, Bi-moteur, Mode Test,
    # Shutdown matériel et Réception LoRa.
    #fonctionne avec console_V10
"""
Branchements (Pins Physiques) :

    Moteur (ESC) : Pin 12 (GPIO 18 - PWM).

    Interrupteur TEST : Pin 13 (GPIO 27) relié au 3.3V (Pin 1).
    LED témoin : Pin 37 (GPIO 26)

    Bouton Shutdown : Pin 5 (GPIO 3) relié au GND (Pin 9).

    GPS : Pin 8 (TX Pi -> RX GPS) et Pin 10 (RX Pi -> TX GPS)
          VCC à 5V et masse commune.

    LoRa : SPI standard + Pin 22 (Reset) + Pin 7 (DIO0).
    Pin Ra-02,Rôle,Pin Raspberry (Physique),GPIO
        VCCLoRa  ->  Pin 1 (ou 17)(3.3V jamais 5V) : alimentation
        GNDLora  -> Pin 6 (ou 9, 14, 20...)": Ground
        NSS (CS) -> Pin 24,GPIO 8 (CE0) : Chip Select
        MOSI -> Data In,Pin 19,GPIO 10
        MISO -> Pin 21,GPIO 9 : Data Out
        SCK -> Pin 23,GPIO 11 : Horloge
        RESET -> Pin 22,GPIO 25 : Réinitialisation
        DIO0 -> Pin 7,GPIO 4 : Interrupt

    Moteur GAUCHE (ESC) : Signal sur GPIO 18 (Pin 12).
    Moteur DROIT (ESC) : Signal sur GPIO 13 (Pin 33).

# Activation du bouton Shutdown sur GPIO 3
    dans /boot/config.txt
    echo "dtoverlay=gpio-shutdown" | sudo tee -a /boot/config.txt

# Installation des dépendances
    sudo apt update && sudo apt install -y python3-serial python3-pip
    pip3 install spidev

"""

import serial, time, math, spidev, os
import RPi.GPIO as GPIO

# --- CONFIGURATION DES PINS ---
ID_BOUEE = "B01"
PIN_MOT_G = 18    # Moteur Gauche
PIN_MOT_D = 13    # Moteur Droit
PIN_LED = 26      # LED Statut
```

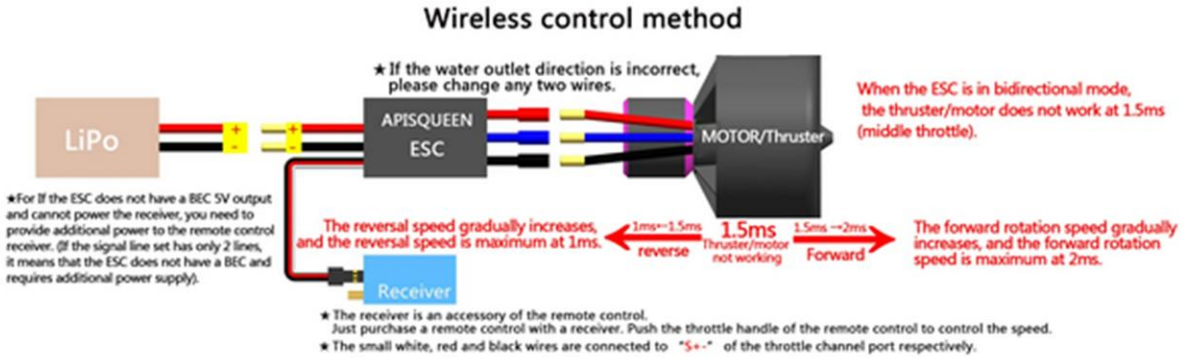
5. Phase de tests :

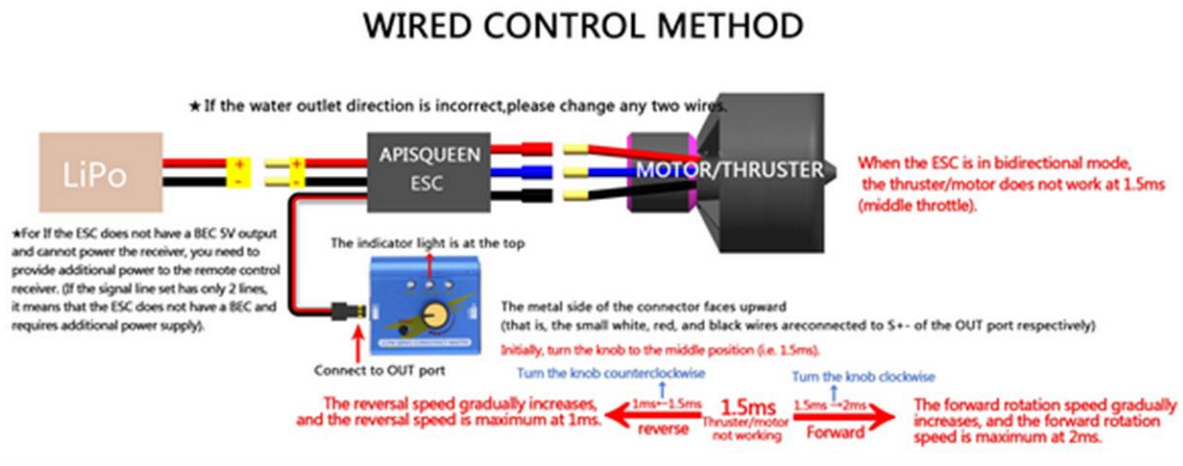
Voici ce que la carte GPS capture et envoie à la console :

(Ici avec la version 20 de la bouée. On constate un écart d'environ 6 à 8m avec la réalité, cependant nous sommes dans le couloir donc le toit du bâtiment gène la réception)

```
Bouée B01 V20 Démarrée.
msg_bytes = b'B01;47.03960;6.27450;50;30;70'
Émis : B01;47.03960;6.27450;50;30;70
msg_bytes = b'B01;47.03960;6.27450;50;30;70'
Émis : B01;47.03960;6.27450;50;30;70
msg_bytes = b'B01;47.03960;6.27450;50;30;70'
Émis : B01;47.03960;6.27450;50;30;70
```

Les moteurs et leur système de commande ont nécessité de nouvelles recherches.





Exemple partiel de la procédure de test des moteurs donnée par l'IA Grok

- **Préparation matérielle (avant de lancer le script) : Débranche tout** : Pas de batterie sur l'ESC pour l'instant.
- **Branchements** (rappel / confirmation) :
- Signal PWM gauche : Pin 12 physique (GPIO 18) → fil signal blanc/jaune de l'ESC gauche
- Signal PWM droit : Pin 33 physique (GPIO 13) → fil signal de l'ESC droit
- GND commun : Relie GND Pi (ex. Pin 6) au GND de l'ESC (noir)
- Alim ESC : Batterie LiPo / source 12-24V (3-6S) sur les fils rouge/noir épais de l'ESC (pas sur le Pi !)
- Moteurs : Les 3 fils brushless de chaque moteur connectés à l'ESC correspondant (ordre quelconque au début ; si rotation inverse, inverse 2 fils).

- **Sécurité** :
- **Pas d'hélice / propulseur monté pour le test (ou fixe-les solidement si tu veux voir la rotation).**
- **Garde la bouée immobile / surélevée.**
- **Aie un moyen d'arrêter vite (débrancher batterie).**

- **Test 1 : Neutre + arming basique (sans calibration)**
- Mets SIMULATION_MODE = True ou laisse le GPS comme ça.
- Lance le script tel quel (python3 BoueeV20.py).
- Observe :
- Au démarrage : les PWM démarrent à 7.5 % → neutre.
- Branche la batterie sur les ESC → tu devrais entendre :
- Quelques beeps initiaux (nombre de cellules détectées, ex. 3 beeps pour 3S).
- Puis un ou deux beeps "arming" si neutre est détecté.
- Pas de rotation des moteurs.
- Si beeps continus ou rien : débranche, attends 10 s, rebranche (certains ESC reset à chaque power-on).

Intégration des éléments de mesures:

(En parallèle de l'intégration des modules LoRA et des moteurs)

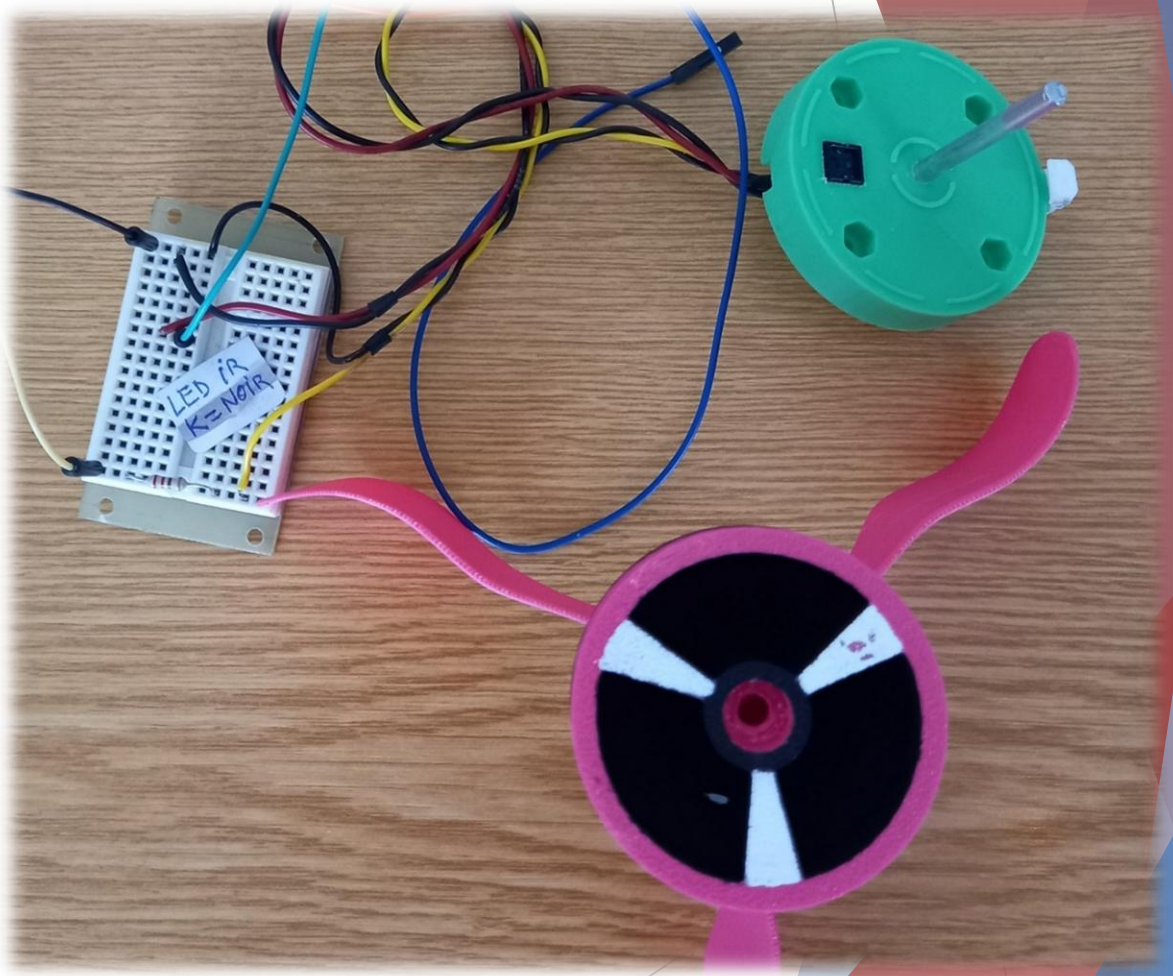
- L'anémomètre a été intégré assez vite à la bouée car il ne nécessite pas de ressource particulière (il n'utilise ni bus ni registres spécifiques donc il ne peut pas rentrer en conflit avec un autre élément) Il faudra seulement l'étalonner.
- Le système LoRa a été très long à intégrer car sa programmation est complexe et il est sensible aux parasites (c'est le code informatique qui doit les éliminer)
- La girouette est venue plus tard car elle nécessite un module boussole qui lui nécessite des ressources communes à d'autres systèmes. De plus nous avons rencontré des difficultés avec ce module.

Fonctionnement de l'anémomètre :

(Pour l'instant câblé sur une platine d'essais)

Lorsque les pâles tournent sous la force du vent, un courant électrique est activé à chaque passage sur les traits blancs en dessous de l'anémomètre (appareil rose).
Il suffit de diviser par 3 le nombre de connexion pour obtenir le nombre de tour.

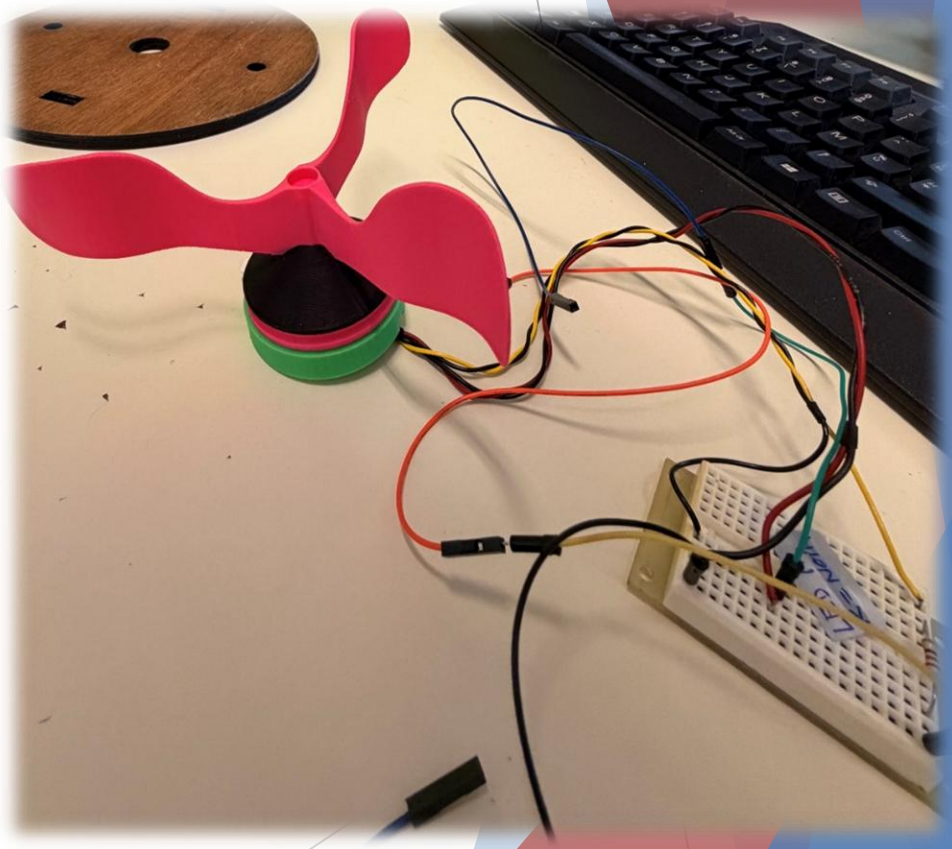
Sa relation avec le temps permettra de connaître la vitesse du vent qui sera transmise à la console. Cependant, à cause, entre autres, des frottements, il faudra appliquer un coefficient multiplicateur. On l'étalonnera donc par comparaison avec un appareil du commerce



Intégration de l'anémomètre au reste de la console informatique

```python
import RPi.GPIO as GPIO
import time
from math import pi

# --- Configuration ---
ANEMOMETRE_PIN = 23    # Broche BCM 23 (Pin physique 16)
TEMPS_MESURE_S = 10.0   # Durée de la mesure en secondes
IMPULSIONS_PAR_TOUR = 3.0
DIAM = 0.13 # diametre moyen de l'anémometre en m
# --- Variables globales ---
compteur_impulsions = 0

def compte_tour(channel):
    """ Fonction de rappel (callback) appelée à chaque impulsion du capteur. """
    global compteur_impulsions
    compteur_impulsions += 1
    print(f"Impulsion détectée ! Total : {compteur_impulsions}") # Pour le débogage

# --- Programme principal ---
try:
    # 1. Configuration du mode
    GPIO.setmode(GPIO.BCM)

    # 2. Configuration de la broche en entrée, avec pull-down pour éviter les faux signaux
    GPIO.setup(ANEMOMETRE_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

    # 3. Ajout de l'interruption : déclenche compte_tour() à chaque front montant
    GPIO.add_event_detect(ANEMOMETRE_PIN, GPIO.RISING, callback=compte_tour, bouncetime=50) # bouncetime pour éviter le rebond

    print("--- Test Anémomètre ---")
    print(f"Mesure démarrée sur GPIO {ANEMOMETRE_PIN} pendant {TEMPS_MESURE_S} secondes. Faites tourner les pales !")

    # Attente pour la mesure
    start_time = time.time()
    time.sleep(TEMPS_MESURE_S)
    end_time = time.time()

    # Suppression de l'interruption pour terminer la mesure
    GPIO.remove_event_detect(ANEMOMETRE_PIN)

    # 4. Calcul des résultats
    temps_reel = end_time - start_time

    if compteur_impulsions > 0:
        tours = compteur_impulsions / IMPULSIONS_PAR_TOUR
        rpm = (tours / temps_reel) * 60.0
        v = tours * DIAM * pi / TEMPS_MESURE_S

        # Le facteur d'étalonnage réel (m/s par RPM) sera ajouté plus tard.
```

Voici, à droite, une image de l'anémomètre tandis qu'à gauche se trouve son programme, prêt à l'utilisation.



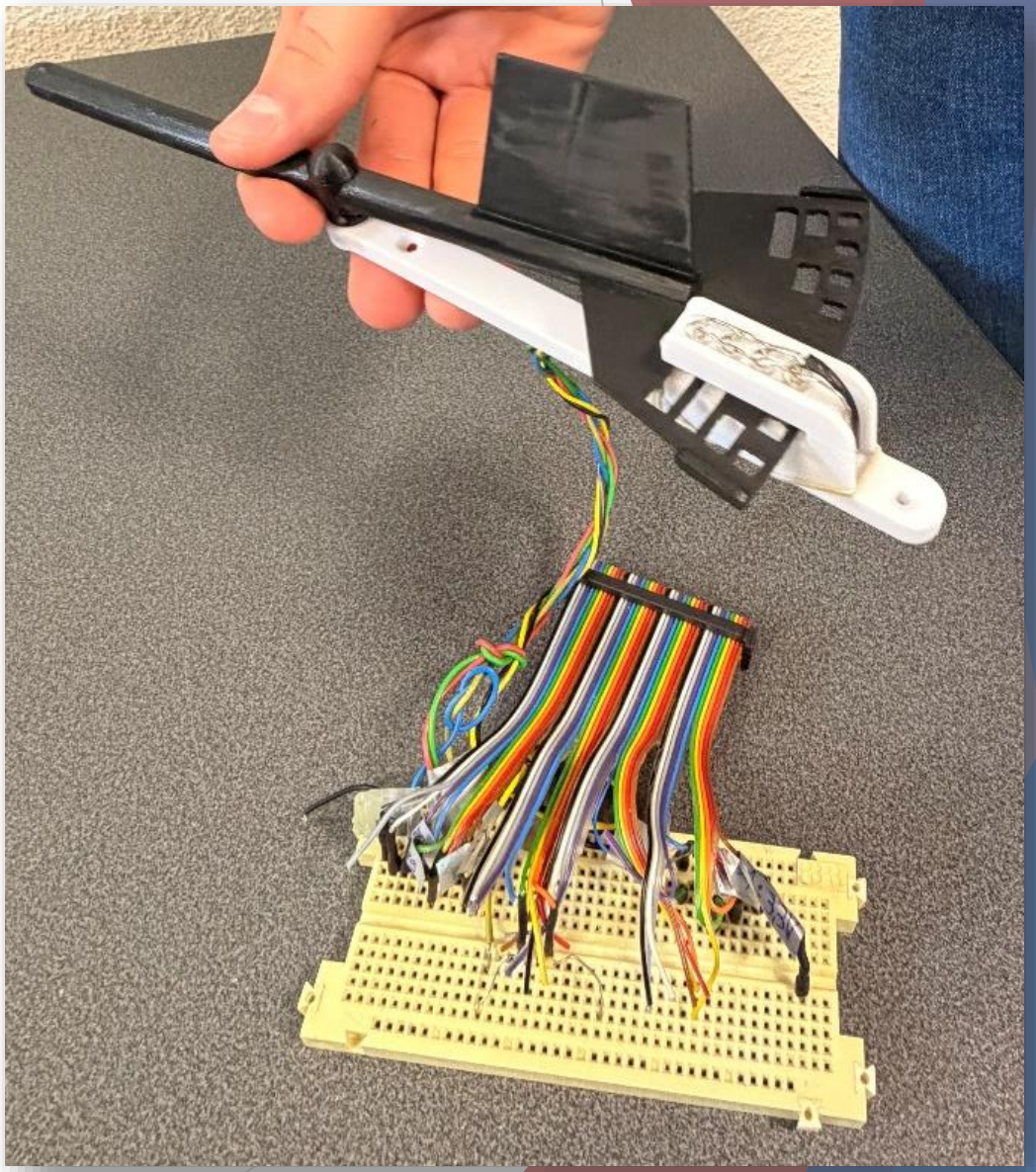
Fonctionnement de la girouette :

Imaginons que la girouette, sous l'impulsion du vent dévie de sa position initiale.

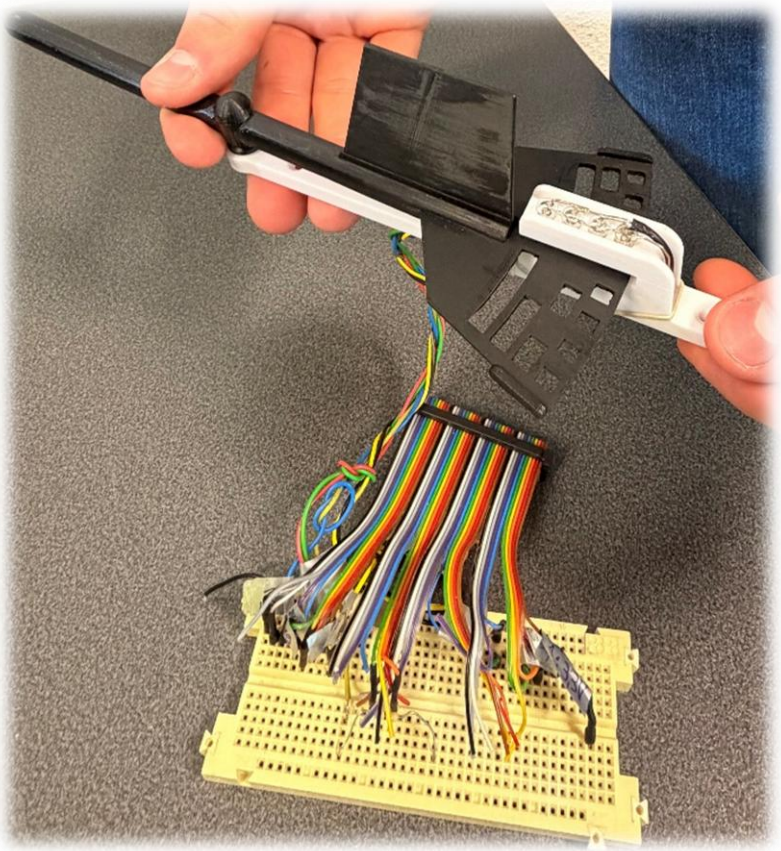
Un signal lumineux est envoyé en permanence par les quatre DEL au-dessus de la partie noire de la girouette. La lumière passant par les trous de cette partie noire est captée par quatre phototransistors ce qui crée un signal en binaire allant de 0 à 15.

Le microcontrôleur le capte et agit sur les moteurs pour placer la bouée face au vent.

Le fait de s'autoriser un débattement de 0 à 15 nous permettra de régler la précision de notre placement.



Intégration de la girouette…



Voici, à gauche, une image de la girouette tandis qu'à droite se trouve son programme, prêt à l'utilisation.

```python
import RPi.GPIO as GPIO
import time

# --- Configuration des Broches BCM ---
# Bit 0 (le plus faible, pour la magnitude 2°/4°)
BIT0_PIN = 26  #(broche physique 37)
# Bit 1 (magnitude 4°/8°)
BIT1_PIN = 19 #(broche physique 35)
# Bit 2 (magnitude 8°/16°)
BIT2_PIN = 13 #(broche physique 33)
# Bit 3 (le plus fort, pour la direction Babord/Tribord)
DIRECTION_PIN = 6 #(broche physique 31)

# Liste des broches pour faciliter la lecture
PINS = [DIRECTION_PIN, BIT2_PIN, BIT1_PIN, BIT0_PIN]
PIN_NAMES = ["Direction (Bit 3)", "Mag. 8° (Bit 2)", "Mag. 4° (Bit 1)", "Mag. 2° (Bit 0)"]

# --- Définition des Plages d'Angles (selon votre spécification) ---
# Le dictionnaire mappe le code binaire (en int) à la plage d'angle moyenne et au texte
# Note : 0000 (0) est spécial.
ANGLE_MAP = {
    # Magnitude (Bits 2, 1, 0) : Plage (moyenne en degrés)
    1: 3.0,  # 001 (2° à 4° -> 3.0°)
    2: 5.0,  # 010 (4° à 6° -> 5.0°)
    3: 7.0,  # 011 (6° à 8° -> 7.0°)
    4: 9.0,  # 100 (8° à 10° -> 9.0°)
    5: 11.0, # 101 (10° à 12° -> 11.0°)
    6: 13.0, # 110 (12° à 14° -> 13.0°)
    7: 15.0, # 111 (14° à 16° -> 15.0°)
}

def lire_girouette():
    """ Lit l'état de chaque broche et retourne l'angle et la direction. """

    # Lecture des états. GPIO.HIGH est VRAI (1) car nous utilisons PULL_UP
    # MAIS : le phototransistor ÉTEINT (coupé) laisse la broche en HIGH (1)
    # Le phototransistor ALLUMÉ (trou) tire la broche vers GND, donc LOW (0)

    # Nous inversons la lecture pour que '1' signifie 'Trou détecté/Signal ON'

    bit_values = []

    for pin in PINS:
        # **Hypothèse la plus probable pour un codeur optique : Trou détecté = 0 (LOW)**
        # Donc : 1 (HIGH) si le trou n'est PAS là (obstrué) et 0 (LOW) si trou détecté
        bit_state = GPIO.input(pin) # lit 1 cache ou 0 trou c'est l'inverse de ce que l'on veut
        # inversion logique
        if bit_state == 1:
            bit_state = 0
```

Réalisation de tests de communication

Voici une des parties du protocole de tests proposés par l'IA « Gemini » afin d'assurer la bonne communication entre le module GPS et la console

Le Diagnostic Final : Le "Bus Fantôme"

Si vous recevez la même suite d'octets peu importe ce que la bouée envoie (et même si elle n'envoie rien), cela signifie que **la console ne lit pas la radio**. Elle lit un "écho" électrique présent sur ses propres fils.

Voici pourquoi :

1. **L'écho du SPI :** Sur le Raspberry Pi 4, si le fil **MISO** (celui qui reçoit les données) est mal connecté, trop long, ou s'il y a un problème de masse (GND), il finit par ramasser par induction le signal de l'horloge (SCK) ou du MOSI.
2. **Le motif 0x23 0xAD 0xA5 :** Ce n'est pas du texte, c'est la signature d'un bus SPI qui "boucle". C'est le Raspberry qui lit ses propres commandes avec un décalage de quelques micro-secondes.
3. **Le décalage de 1 bit :** Si vous regardez bien, `0x23` ressemble mathématiquement à une version décalée de l'adresse de registre que le Pi envoie.

Le Test de Vérité (Hardware Check)

Pour confirmer cela une bonne fois pour toutes, nous allons demander au Raspberry de lire un registre dont la valeur est **fixe et connue d'avance** (le registre de version de la puce).

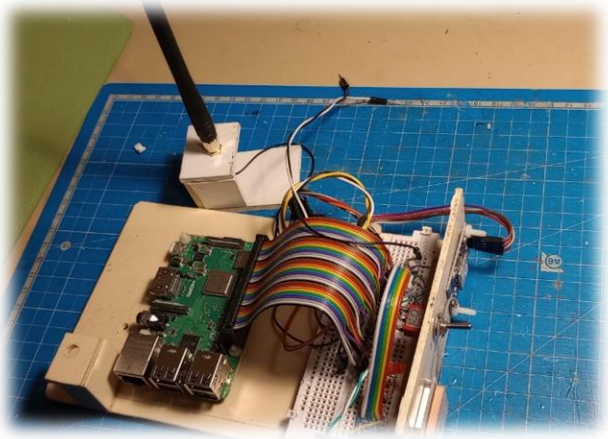
Que faire selon le résultat ?

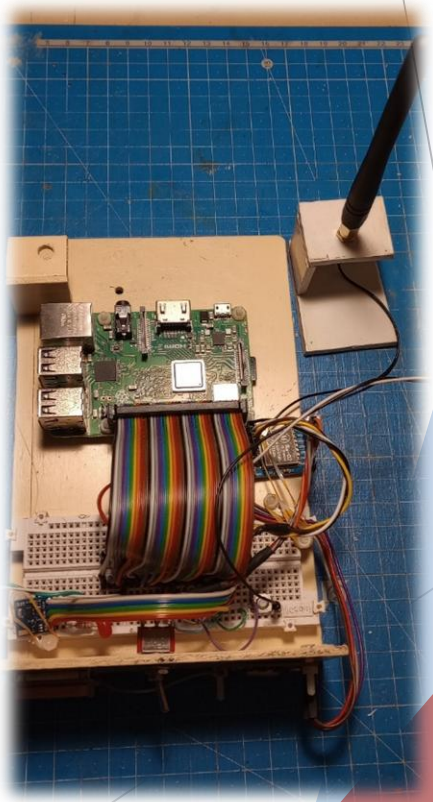
- **Si vous obtenez `0x12` :** Alors mon diagnostic de "fantôme" est faux, et c'est la configuration radio (fréquence/syncword) qui est décalée.
- **Si vous obtenez n'importe quoi d'autre (comme `0x23` ou `0xAD`) :** C'est la preuve que le signal électrique est "sale".

La carte « console ».
Basée sur un raspberry 4B+, elle reçoit les données, les interprète et les affiche à l'écran.
Elle envoie aussi des ordres à la bouée.

Initialement elle était basée sur un RP3b+ comme la bouée mais il s'est avéré trop peu puissant pour gérer la réception des données en plus de l'affichage.

Juste en dessous de la nappe de fils multicolores on aperçoit le module Lora. Tout en bas de l'image on voit son antenne.





La carte « Bouée »,
Basée sur un raspberry pi 3B+, capture les données et les envoie à la console. Elle pilote aussi les moteurs.
Elle sera placée verticalement. Ainsi l'interrupteur et l'antenne GPS seront en haut. L'antenne LoRa sortira sur le coté.

6. Début de l'assemblage de la bouée :

Les opérations commencent par la fixation des moteurs sur la plaque de base…

7. Correction du défaut de réception lié à l'atténuation du signal à l'intérieur de la bouée

(En parallèle des tests)

Nous constatons que l'émission du signal est atténuée par les chambres à air en caoutchouc noir.
En effet, celui-ci contient beaucoup de carbone qui est très conducteur c'est probablement ce qui atténue les ondes.
En attendant de fixer différemment l'antenne, nous faisons les tests avec une bouée partiellement démontée.

… ET POSE DU REVÊTEMENT, UNE BÂCHE IMPERMÉABLE JAUNE FLUO.

8. Finalisation du revêtement, partiellement réalisé

L’étape suivante sera de mettre les deux petites bouées au-dessus des deux grandes de manière à recouvrir la partie en bois qui abritera la boite étanche de l’électronique.

Conclusion

Nous avions prévu un certain nombre d'éléments sur cette bouée géopositionnée, comme de pouvoir déplacer les bouées depuis l'écran de contrôle par des clics ou avec un joystick. De plus, chaque bouée doit transmettre sa position (coordonnées GPS), la vitesse, la direction du vent et l'état de sa batterie, à la console de commande. La console doit également afficher une perte éventuelle de contact avec la bouée.

La plupart des éléments ont été réalisé, seules, la direction du vent, l'état de la batterie et la commande par joystick ont été remis à plus tard, soit par manque de temps soit, a cause d'une panne d'un élément. De même les tests sur l'eau attendront une météo plus clémente

En plus de l'informatique de base, ce projet nous a appris beaucoup d'autres choses. Nous avons appris à utiliser un microcontrôleur de façon beaucoup plus complète que ce que prévoit le programme de la spécialité NSI (gestion des bus, etc) et nous avons appliqué nos cours de programmation sur des cas concrets.

Ainsi nous avons compris que le travail d'un programmeur est avant tout basé sur la compréhension du problème a résoudre. Même l'IA a besoin qu'on lui décrive précisément ce que l'on veut qu'elle fasse, sans cela elle répond souvent a coté.

Nous avons aussi beaucoup appris sur le travail en équipe, la gestion des rôles et le partage des tâches. Ce projet a enfin été une épreuve de patience et une école de persévérance car la programmation des bus du raspberry est souvent liée à des détails et vu que tout est codé en hexadécimal, ce n'est pas simple à dépanner.