



OLYMPIADES
DE PHYSIQUE FRANCE



Annexe du Mémoire

Titre du projet :

Construction d'une fusée de A à Z

Élaboré par :

Enzo Sanchez Sartori

&

Elliot Matton–Marie

Avec l'aide du professeur :

Nicolas Debons

Sommaire

Sommaire.....	1
Annexe 1 – Projets personnels	2-4
Annexe 1.1 – Projets d’Enzo.....	2
Annexe 1.2 – Projet d’Elliot.....	3
Annexe 1.3 – Explication du logo de notre projet	4
Annexe 2 – Nova 01 tiny.....	5
Annexe 3 – Problèmes rencontrés	6-7
Annexe 3.1 – Avec le système de récupération	6
Annexe 3.2 – Avec le tiror	7
Annexe 4 – Expérience du lancer de parachute.....	8
Annexe 5 – Matériel, consommables et logiciels.....	9
Annexe 5.1 – Matériel.....	9
Annexe 5.2 – Consommables.....	9
Annexe 5.3 – Logiciels	9
Annexe 6 – Ordinateur de bord	10
Annexe 6.1 – Schéma papier des connexions	10
Annexe 6.2 – Expérience de l’étalonnage du capteur BMP280.....	11-12
Annexe 6.3 – Code Python.....	135

Annexe 1 – Projets personnels

Annexe 1.1 – Projets d'Enzo

Le projet de voiture radiocommandée d'Enzo a pour nom « Pulsar ». Plusieurs photos de ce projet ainsi que le logo du nom du projet sont présentés en *Figure 1*.

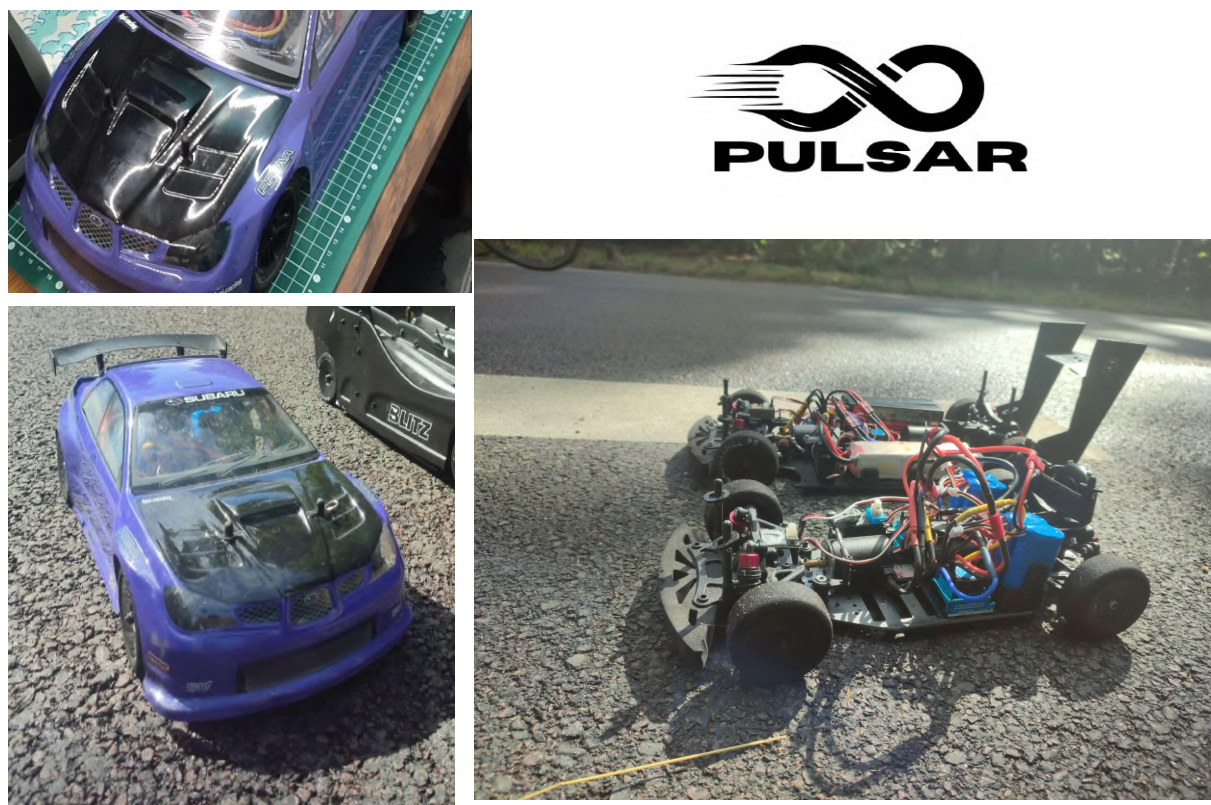


Figure 1 : Trois photos de la voiture radiocommandée développée par Enzo, ainsi que le logo du projet « Pulsar ».

La vitesse maximale atteinte par cette voiture à ce jour est de 70 km/h. Le but est d'atteindre la vitesse symbolique des 100 km/h, en optimisant l'espace occupé et en conservant les dimensions d'origine.

Le circuit électrique est composé de 8 cellules électriques destinées à donner 14.8V. Il y a également deux moteurs qui tournent à 50.000 tours minute théoriquement. La vitesse de 100 km/h est *a priori* dépassée mais les tests sont compliqués à réaliser à de telles vitesses.

Enzo a également réalisé plusieurs petits projets trop courts pour être expliqués :

- Modélisation 3D d'un bras de robot (première image de la *Figure 2*) ;
- Restauration de voiture 1/24 ;
- Réaménagement de chambre en la modélisant en 3D (deuxième image de la *Figure 2*) ;

- Construction d'une propre tour PC (avec l'aide d'Elliot) ;
 - Réalisation d'une « Trokart », mélange entre une voiture de karting et une trottinette électrique (troisième image de la *Figure 2*) ;
- Restauration d'une Pocket bike (une très petite moto).

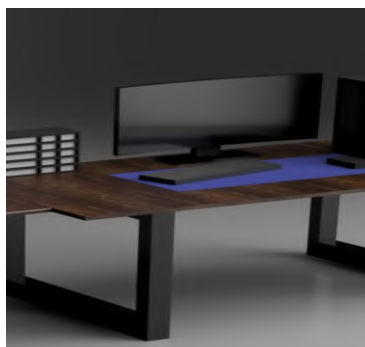
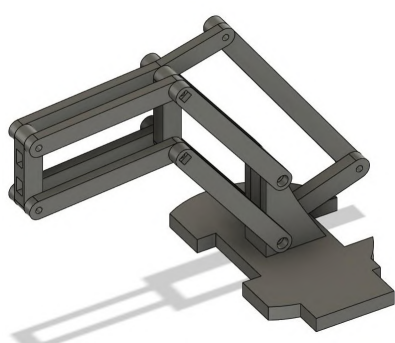


Figure 2 : Trois images de mini-projets menés par Enzo. Image de gauche : modélisation 3D d'un bras de robot. Image centrale : modélisation 3D d'une chambre avec notamment le bureau. Image de droite : photo d'une « Trokart ».

Annexe 1.2 – Projet d'Elliot

Le projet de voiture radiocommandée d'Elliot a pour nom « Endurance » car il y a besoin d'endurance financière et d'endurance mentale dans ce genre de projet.

Elliot a réalisé sa voiture en agrandissant le châssis pour accueillir plus de composants. Il a dû modéliser un nouveau châssis en fibres de carbone. Il a par la suite commandé une plaque en fibres de carbone qu'il a découpée et percée.

Nous sommes tous les deux en pause sur ce projet de voiture radiocommandée. Nous reprendrons les tests une fois les conditions expérimentales plus favorables, donc en printemps/été (il faut notamment un sol un minimum chaud pour améliorer l'adhérence).



Figure 3 : Image du projet d'avion motorisé en bois mené par Elliot.

Annexe 1.3 – Explication du logo de notre projet

Enzo utilise le nom de « Pulsar » car c'est une étoile à neutrons qui tourne extrêmement rapidement sur elle-même. Ses projets ont souvent un lien avec la vitesse et Enzo aime beaucoup le symbole comme le nom (voir *Figure 4*).

Elliot utilise le nom de « AEN » qui est l'acronyme de « Air, Endurance, Nova ». Ce sont les noms de ses trois plus gros projets à ce jour. A pour Air, un avion motorisé en balsa (type de bois très léger) de 2 mètres d'envergure ; E pour Endurance, la voiture radiocommandée ; N pour NOVA (voir *Figure 4*).



Figure 4 : Logo de notre projet NOVA : « Construction d'une fusée de A à Z ».

Annexe 2 – Nova 01 tiny

Cette mini-fusée entièrement réalisée par impression 3D a été réalisée dans un premier temps pour nous initier au domaine de la fuséologie. Elle s'appelle Nova 01 tiny (voir *Figure 5*). C'est pourquoi nous souhaitons un moment que la fusée qui est l'objet de ce projet soit nommée Nova 2E ou Nova EE (deuxième tentative de réalisation du projet de construction de fusée).



Figure 5 : Photo de notre première fusée NOVA 01 parée au décollage.

Annexe 3 – Problèmes rencontrés

Annexe 3.1 – Avec le système de récupération

Nous avons rencontré les problèmes suivants durant la modélisation du système de récupération :

- La Base une fois imprimée, nous nous sommes rendu compte que ses dimensions avaient des valeurs ne permettant pas le bon emboîtement des pièces au niveau du rail en métal. Nous avons donc dû apporter des modifications et la réimprimer.
- la résistance mécanique de la partie rouge de la Base était faible donc nous avons dû apporter de nouveau des modifications avant de la réimprimer.
- La partie Amovible (en vert) était sujette à la flexion vers l'intérieur ce qui empêchait les *triggers* de bien s'enlever. Nous avons donc dû créer une nouvelle pièce pour empêcher le mouvement des pièces de la partie Amovible.

Nous avons fait des tests de résistance mécanique en accrochant la totalité du système assemblé à une masse. Cela nous a permis de tester avec succès la bonne jointure du système (voir *Figure 6*).



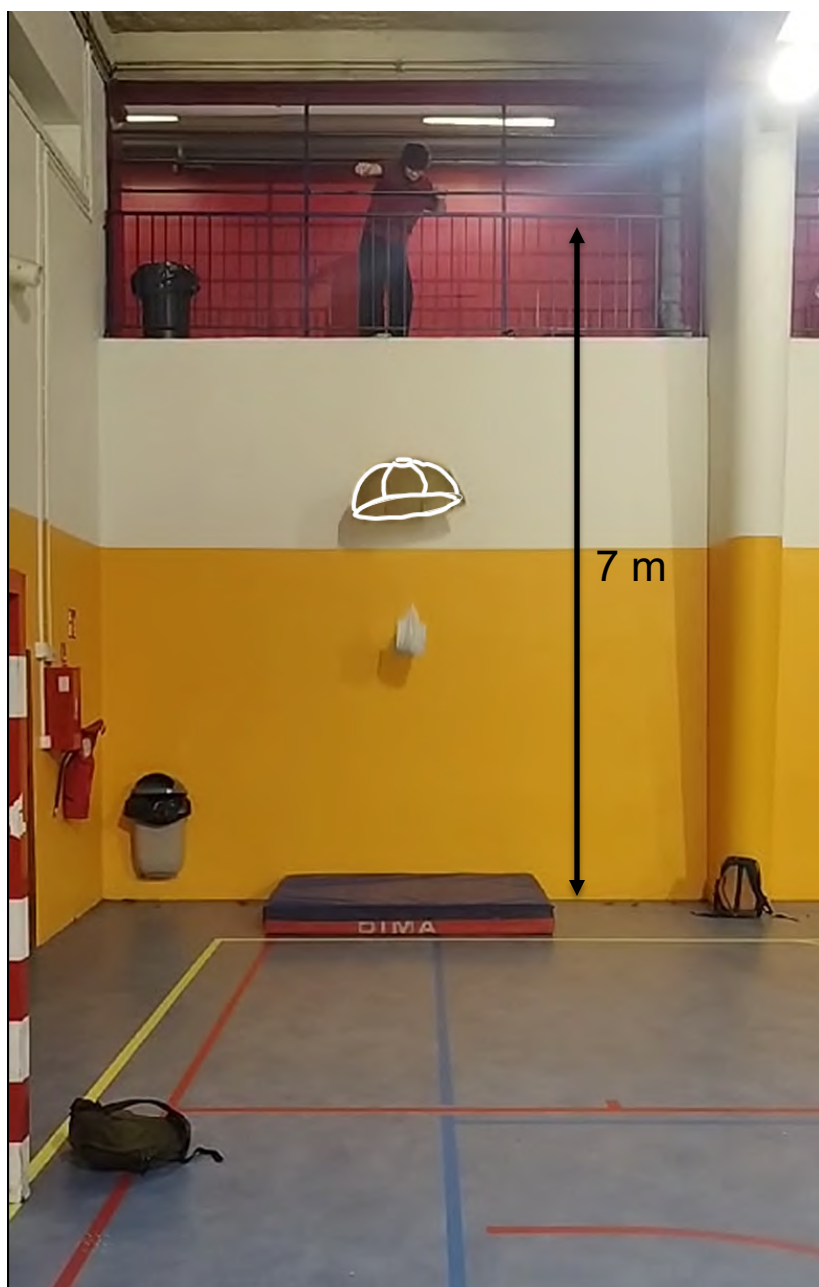
Figure 6 : Photos du système de récupération final.

Annexe 3.2 – Avec le tiroir

La première version de fermeture du tiroir comprenait un ressort qui permettait à la poignée de rester collée à son socle. Lorsqu'il fallait tourner la poignée, on devait d'abord la tirer vers le haut, ce qui compressait le ressort. Ce système apportait des problèmes au niveau de l'ouverture car on s'est aperçus que le tiroir ne s'ouvrait plus.

Cette idée a donc été écartée pour finalement conserver le système sans le ressort.

Annexe 4 - Expérience du lancer de parachute



Après avoir construit notre parachute nous avons testé sa capacité à se déployer lors de la phase de chute pour assurer le freinage de la fusée et éviter de casser la fusée.

C'est dans cette optique que l'on a réalisé une expérience dans le gymnase de notre lycée. C'est dans ce lieu que nous avons trouvé la plus grande hauteur (7 m) pour réaliser la chute. Cette expérience nous a permis de valider l'ouverture du parachute et sa forme hémisphérique lors de la chute (voir *Figure 7*).

Nous aurions aimé pouvoir exploiter la vidéo et obtenir à l'aide d'une chronophotographie la vitesse maximale atteinte par le système {parachute + masse} au cours de la chute. Malheureusement la hauteur n'était pas suffisante pour que le système atteigne une vitesse constante. Nous aimerions prochainement pouvoir réaliser à nouveau cette expérience avec une hauteur de chute plus grande.

Figure 7 : Image extraite de la vidéo obtenue lors de l'expérience de lancer de parachute.

Annexe 5 – Matériel, consommables et logiciels

Annexe 5.1 – Matériel

Outils :

Pied à coulisse / règle/ tournevis / clé à laine / balance / calculatrice / papier de verre / visseuse / cutter / scie / multimètre / fer à souder.

Machines :

Ordinateur fixe et portable / BambuLab A1 (imprimante 3D) / imprimante papier.

Annexe 5.2 – Consommables

Quincaillerie :

Vis M3 (de 5mm à 30mm) / ressort.

Matériaux :

PETG / carton / aluminium / composants électroniques.

Annexe 5.3 – Logiciels

Pour la 3D “Fusion 360” :

<https://www.autodesk.com/ca-fr/products/fusion-360/overview?term=1-YEAR&tab=subscription>

Pour l’ordinateur (ARC) “Easy EDA” :

<https://easyeda.com/fr>

Pour créer les fichier lisibles par l'imprimante 3D “Bambu studio” :

<https://bambulab.com/en/download/studio>

Pour conception théorique d’une fusée “Openrocket” :

<https://openrocket.info/>

Pour le code “Visual studio” :

<https://code.visualstudio.com>

Annexe 6 – Ordinateur de bord

Annexe 6.1 – Schéma papier des connexions

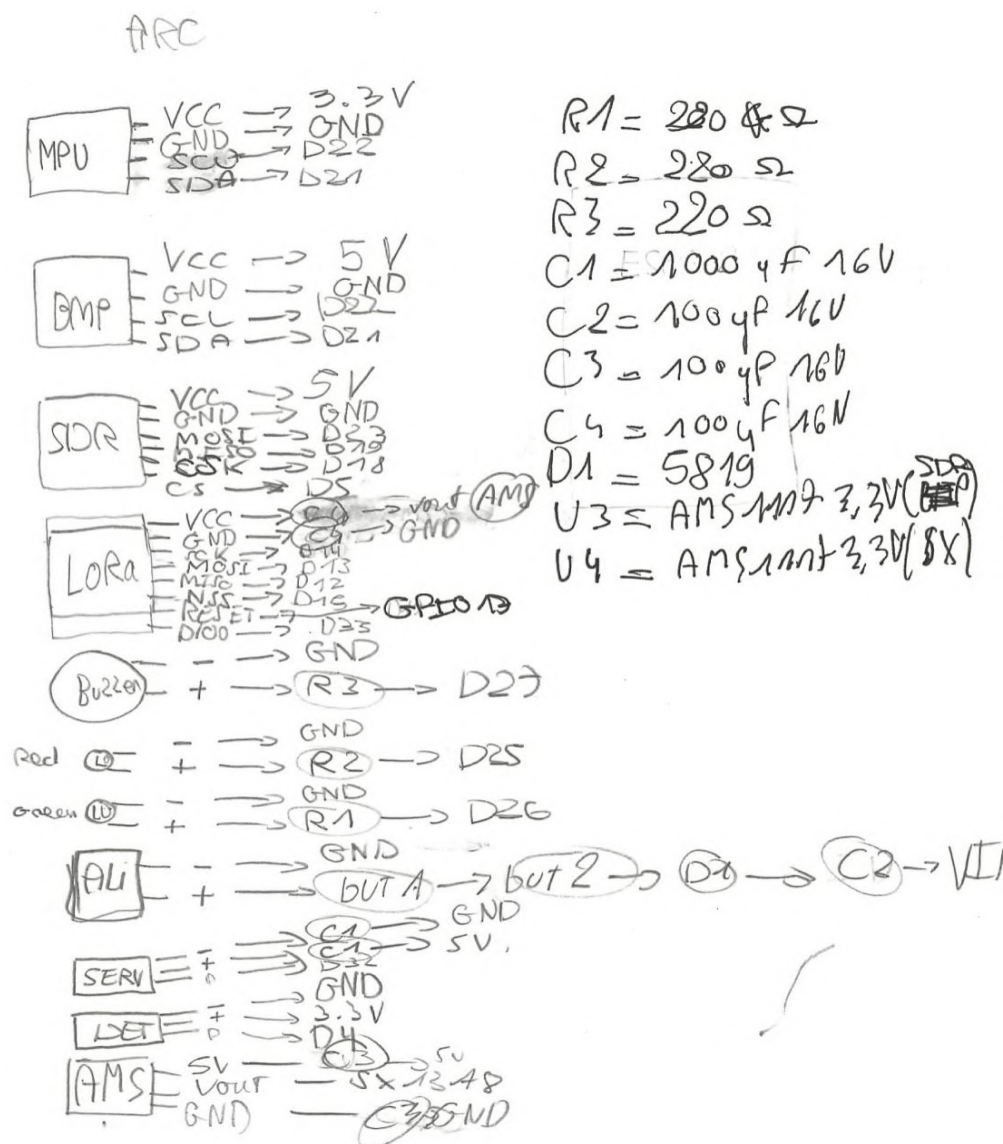


Figure 8 : Schéma papier des connexions de l'ordinateur de bord ARC.

Annexe 6.2 - Expérience de l'étalonnage du BMP280

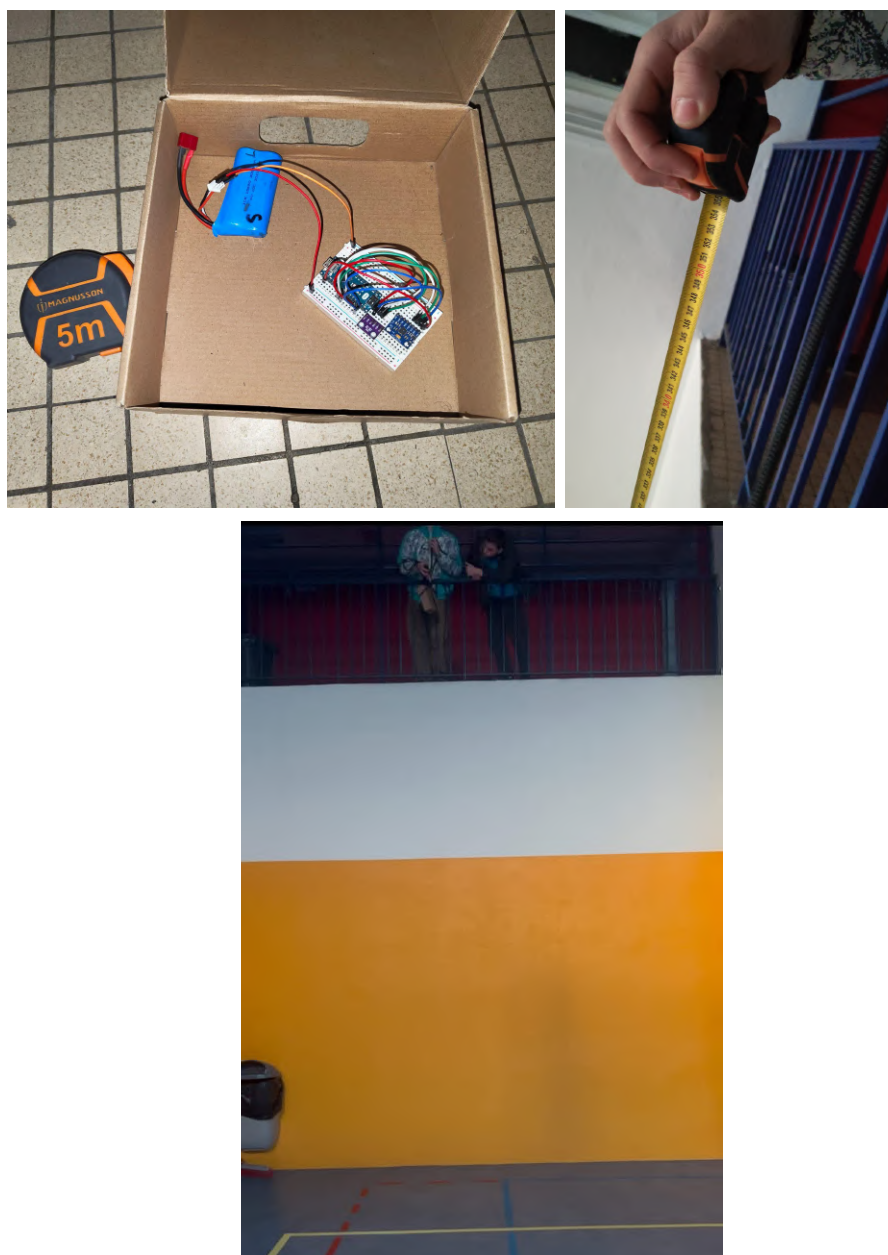


Figure 9 : Photos prises lors de l'étalonnage du capteur barométrique, montrant la boîte contenant le BMP280, le mètre et la hauteur considérée au gymnase du lycée.

Figure 10 : Mesures (temps, pression, altitude) obtenues via le BMP280 lors de l'étalonnage du capteur barométrique. Sont également figurées les mesures réelles d'altitude réalisées à l'aide du mètre.

Temps (s)	Pression (hPa)	Altitude (m)	Altitude réelle (m)
0.0	1013.24	0.02	0
0.1	1013.24	0.05	
0.2	1013.24	0.0	
0.3	1013.22	0.19	
0.4	1013.22	0.22	
0.5	1013.22	0.22	
0.6	1013.2	0.4	
0.7	1013.21	0.28	
0.8	1013.23	0.14	
0.9	1013.2	0.39	
1.0	1013.18	0.52	
1.1	1013.18	0.55	
1.2	1013.17	0.63	
1.3	1013.17	0.66	
1.4	1013.17	0.62	
1.5	1013.17	0.62	
1.6	1013.16	0.73	1
1.7	1013.13	0.99	
1.8	1013.14	0.86	
1.9	1013.12	1.03	
2.0	1013.14	0.87	
2.1	1013.11	1.16	
2.2	1013.11	1.13	
2.3	1013.11	1.16	
2.4	1013.09	1.25	
2.5	1013.08	1.34	
2.6	1013.08	1.34	
2.7	1013.08	1.36	
2.8	1013.07	1.43	
2.9	1013.06	1.49	
3.0	1013.06	1.52	
3.1	1013.06	1.49	
3.2	1013.06	1.5	
3.3	1013.06	1.55	
3.4	1013.04	1.71	
3.5	1013.04	1.69	
3.6	1013.04	1.68	
3.7	1013.02	1.84	2
3.8	1013.01	1.93	
3.9	1013.01	1.93	
4.0	1013.01	1.94	
4.1	1013.0	2.03	
4.2	1012.99	2.1	
4.3	1013.0	2.05	
4.4	1012.97	2.32	
4.5	1012.99	2.11	
4.6	1012.96	2.41	
4.7	1012.97	2.26	
4.8	1012.94	2.5	
4.9	1012.96	2.4	
5.0	1012.95	2.48	
5.1	1012.94	2.54	
5.2	1012.94	2.5	
5.3	1012.93	2.64	
5.4	1012.92	2.71	
5.5	1012.92	2.69	
5.6	1012.92	2.72	
5.7	1012.91	2.82	
5.8	1012.9	2.85	
5.9	1012.88	3.0	
6.0	1012.88	3.06	
6.1	1012.88	3.04	
6.2	1012.86	3.2	
6.3	1012.88	3.0	3
6.4	1012.87	3.09	
6.5	1012.86	3.24	
6.6	1012.86	3.17	
6.7	1012.84	3.38	
6.8	1012.82	3.5	
6.9	1012.84	3.4	
7.0	1012.82	3.5	
7.1	1012.81	3.63	
7.2	1012.81	3.63	
7.3	1012.82	3.57	
7.4	1012.81	3.64	
8.0	1012.76	4.06	
8.1	1012.75	4.13	4
8.2	1012.75	4.08	
8.3	1012.74	4.18	
8.4	1012.76	4.0	
8.5	1012.74	4.19	
8.6	1012.71	4.46	
8.7	1012.73	4.28	
8.8	1012.72	4.38	
8.9	1012.71	4.49	
9.0	1012.7	4.55	
9.1	1012.71	4.46	
9.2	1012.69	4.61	
9.3	1012.69	4.62	
9.4	1012.67	4.77	
9.5	1012.67	4.74	
9.6	1012.66	4.85	
9.7	1012.66	4.86	
9.8	1012.65	4.91	
9.9	1012.65	4.97	5
10.1	1012.64	5.03	
10.2	1012.66	5.1	
10.3	1012.63	5.05	
10.4	1012.64	5.12	

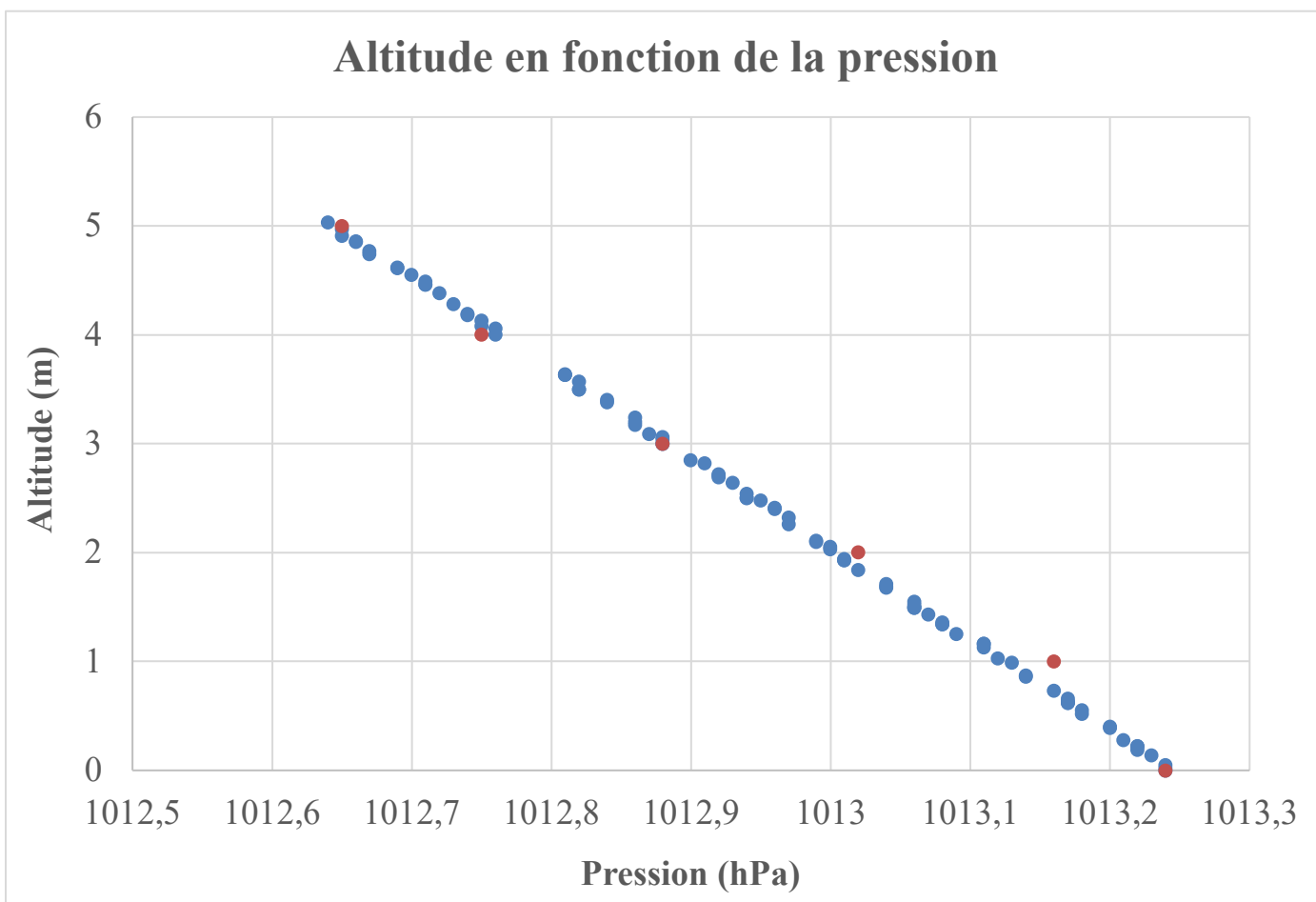


Figure 11 : Graphique montrant l'évolution de l'altitude en fonction de la pression. Les points bleus correspondent aux mesures expérimentales réalisées avec le capteur barométrique, et ceux rouges correspondent aux altitudes mesurées à l'aide du mètre.

Annexe 6.3 – Code C++

```

#include <Wire.h>
#include <SD.h>
#include <SPI.h>
#include <ESP32Servo.h> // Utiliser la bibliothèque ESP32Servo
#include <Adafruit_MPU6050.h>
#include <Adafruit_BMP280.h>
#include <Adafruit_Sensor.h>

// Définitions des broches
#define LED_ROUGE_PIN 25
#define LED_VERT_PIN 26
#define BUZZER_PIN 27
#define SERVO_PIN 32
#define SD_CS_PIN 5
#define USB_DETECT_PIN 4

// Initialisation des capteurs et composants
Adafruit_MPU6050 mpu;
Adafruit_BMP280 bmp;
Servo myServo;
File dataFile;

void detecterProblemes() {
    // Vérifier la carte SD
    if (!SD.begin(SD_CS_PIN)) {
        Serial.println("Erreur de carte SD !");
        digitalWrite(LED_ROUGE_PIN, HIGH);
        digitalWrite(LED_VERT_PIN, LOW); while (1);
    }

    // Vérifier le capteur BMP280
    if (!bmp.begin()) {
        Serial.println("Erreur de capteur BMP280 !");
        digitalWrite(LED_ROUGE_PIN, HIGH);
        digitalWrite(LED_VERT_PIN, LOW); while (1);
    }

    // Vérifier le capteur MPU6050
    if (!mpu.begin()) {
        Serial.println("Erreur de capteur MPU6050 !");
        digitalWrite(LED_ROUGE_PIN, HIGH);
        digitalWrite(LED_VERT_PIN, LOW); while (1);
    }

    // Si tout va bien
    digitalWrite(LED_VERT_PIN, HIGH);
    digitalWrite(LED_ROUGE_PIN, LOW);
}

void enregistrerDonneesDeVol() {
    unsigned long startTime = millis();
    bool volTermine = false;
    bool apogeeAtteinte = false;
    float hauteurMax = 0;

```

```

        while (!volTermine) {
            unsigned long currentTime = millis();
            float hauteur = bmp.readAltitude();
            float temp = bmp.readTemperature();
            sensors_event_t a, g, tempEvent;
            mpu.getEvent(&a, &g, &tempEvent);

            // Enregistrement des données
            dataFile.print(currentTime - startTime);
            dataFile.print(",");
            dataFile.print(hauteur);
            dataFile.print(",");
            dataFile.print(a.orientation.pitch);
            dataFile.print(",");
            dataFile.print(a.orientation.roll);
            dataFile.print(",");
            dataFile.print(a.orientation.heading);
            dataFile.print(",");
            dataFile.print(temp);
            dataFile.print(",");

            // Détection de l'apogée if
            (hauteur > hauteurMax) {
                hauteurMax = hauteur;
                apogeeAtteinte = false;
            } else if (!apogeeAtteinte && hauteur < (hauteurMax - 5)) {
                apogeeAtteinte = true;
                dataFile.print("Apogee");
                myServo.write(180); // Activer le servomoteur
                tone(BUZZER_PIN, 1000, 100); // Bip à l'apogée
                delay(4000); // Attendre 4 secondes avant l'activation
                myServo.write(0); // Réinitialiser le servomoteur
            }

            // Vérification de la fin du vol
            if (hauteur < 1 && apogeeAtteinte) {
                volTermine = true;
                tone(BUZZER_PIN, 1000, 3000); // Bip continu à l'atterrissage
                while (true) {
                    tone(BUZZER_PIN, 1000, 1000); // Bip toutes les 3 secondes
                    delay(3000);
                }
            }

            // Enregistrement de l'activation du servomoteur
            if (apogeeAtteinte) {
                dataFile.println("Servo activated normally");
            } else {
                dataFile.println("Servo not activated");
            }

            // Enregistrement des données
            dataFile.flush();
            delay(1000); // Enregistrement toutes les secondes
        }

        // Fin de l'enregistrement des données
        dataFile.close();
        delay(1000); // Attendre 1 seconde après la détection de la fin du vol
    }

```

```

        void setup() {
            Serial.begin(115200);
            pinMode(LED_ROUGE_PIN, OUTPUT);
            pinMode(LED_VERT_PIN, OUTPUT);
            pinMode(BUZZER_PIN, OUTPUT);
            pinMode(USB_DETECT_PIN, INPUT);

            // Initialisation du MPU6050
            if (!mpu.begin()) {
                Serial.println("Impossible de trouver le MPU6050, checkez les connexions !");
                digitalWrite(LED_ROUGE_PIN, HIGH);
                while (1);
            }

            // Initialisation du BMP280
            if (!bmp.begin()) {
                Serial.println("Impossible de trouver le BMP280, checkez les connexions !");
                digitalWrite(LED_ROUGE_PIN, HIGH);
                while (1);
            }

            // Initialisation du servomoteur
            myServo.attach(SERVO_PIN);

            // Initialisation de la carte SD
            if (!SD.begin(SD_CS_PIN)) {
                Serial.println("Impossible d'initialiser la carte SD !");
                digitalWrite(LED_ROUGE_PIN, HIGH);
                while (1);
            }

            digitalWrite(LED_VERT_PIN, HIGH);
            tone(BUZZER_PIN, 1000, 100); // Bip au démarrage

            // Initialisation du fichier de données
            dataFile = SD.open("datalog.txt", FILE_WRITE);
            if (!dataFile) {
                Serial.println("Erreur d'ouverture du fichier de données !");
                digitalWrite(LED_ROUGE_PIN, HIGH);
                while (1);
            }

            // Écrire les en-têtes de données dans le fichier
            dataFile.println("Temps(ms),Hauteur(m),Position(deg),Temp(C),Activation");
        }

        void loop() {
            detecterProblemes();

            // Vérifier la déconnexion USB pour détecter le décollage
            if (digitalRead(USB_DETECT_PIN) == LOW) {
                Serial.println("USB déconnecté, début du vol détecté.");
                // Commencer à enregistrer les données de vol
                enregistrerDonneesDeVol();
            }
        }
    }

```

Figure 12 : Code C++ de l'ordinateur de bord ARC.