Retour d'expérience
Vif argent

Avec Lucien Delbaere,
Lana Freymann--Damien,
Benjamin Manceaux
et l'association ELISA SPACE

Sommaire

I) Présentation du projet Vif argent

Vif Argent, ou la fusée Coca pour les habitués.

Cette fusée a été initialement conçue pour être légère, et quoi de mieux que des canettes pour atteindre cet objectif ? Nous voulions que notre fusée se distingue par son originalité, et nous pensons avoir réussi, car c'est une première dans l'histoire du C'Space.

Le projet a débuté par une phase de brainstorming où nous avons exploré diverses idées de matériaux et de conceptions. Rapidement, l'idée des canettes s'est imposée comme une solution évidente pour allier légèreté et innovation.

Ce projet unique dans l'histoire du C'Space nous a permis de nous démarquer et de capter l'attention des autres participants ainsi que des organisateurs. La fusée Vif Argent, surnommée la fusée Coca, est devenue un symbole de créativité, démontrant que des matériaux simples et recyclés peuvent être utilisés dans des projets de haute technologie.

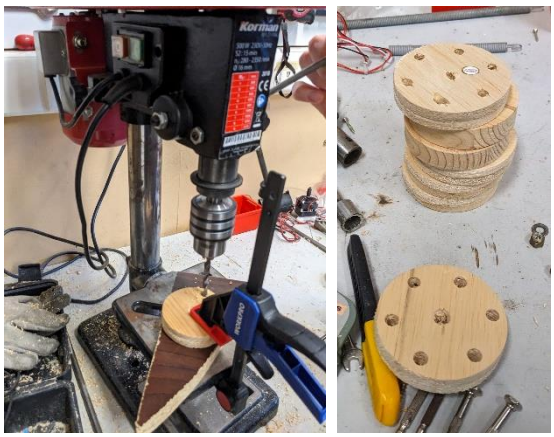
L'utilisation de canettes présentait plusieurs défis, notamment en termes de rigidité structurelle et de résistance aux forces du lancement. Nous avons surmonté ces obstacles en utilisant des techniques de renforcement et en effectuant de nombreux tests pour assurer la fiabilité de notre conception, dont la microfusée de la minifusée :

Photo de la micro de la minif

II) Structure

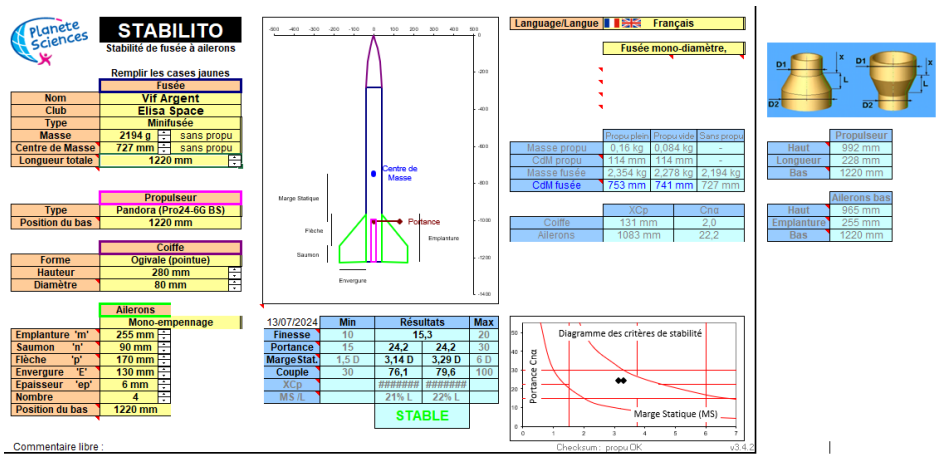
Notre objectif initial était de construire l'intégralité de la fusée en canettes de soda. Cependant, nous avons rencontré des difficultés au niveau de la jonction des canettes. Nous avons tenté d'utiliser un soudeur par points pour les assembler, mais cette méthode s'est avérée inefficace. En effet, l'aluminium des canettes a une épaisseur de 0,33 mm, et la présence de la couche protectrice grise (côté intérieur) ainsi que la couche de peinture aux couleurs de la marque compliquent la soudure. Le soudeur soit n'avait aucun effet sur les canettes, soit l'impulsion transperçait complètement l'aluminium.

Nous avons donc opté pour une structure interne rigide composée de tiges en PVC et de bagues en bois, les canettes servant uniquement à recouvrir cette structure.



Découpe des bagues en bois pour la structure interne

En parallèle de la réflexion sur la structure, il était également nécessaire de prévoir la stabilité de la fusée à l'aide du logiciel Stabtraj. Nous devions essayer de définir des cotes pour la masse, le centre de gravité, la taille des ailerons, le corps de la fusée et de nombreux autres paramètres.

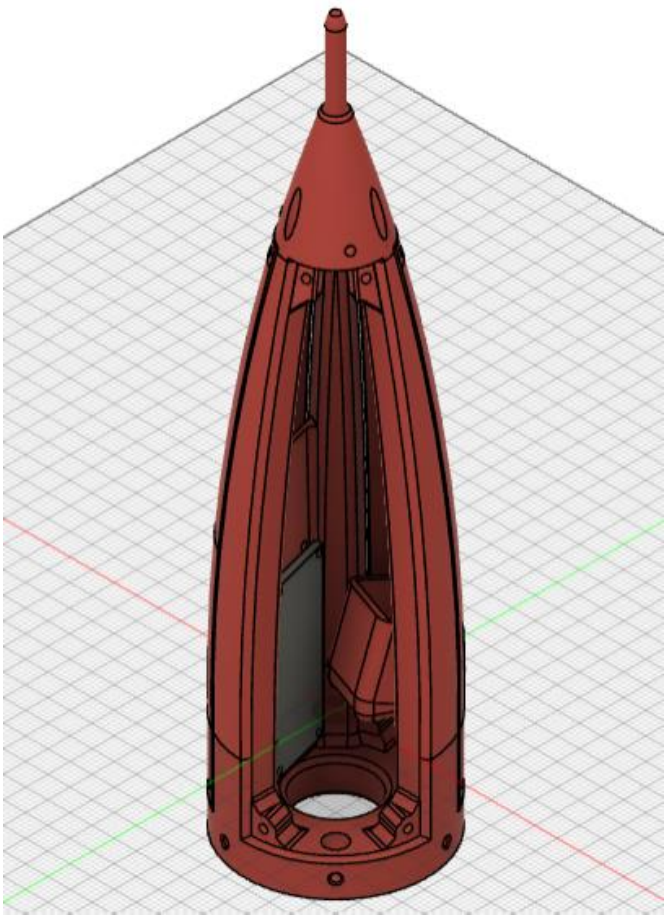


Stabtraj final de VIF ARGENT

Pour réaliser cette fusée stable, nous avons d'abord fixé un diamètre de 80 mm. Ensuite, nous avons déterminé une taille adaptée afin de répondre aux critères de finesse. Nous avons essayé de calculer une masse approximative en utilisant un modèle 3D sur Fusion et en choisissant les matériaux appropriés. Il est possible d'obtenir une estimation de la masse en utilisant la relation $m=\rho \cdot Vm$, en connaissant le volume de chaque pièce et sa masse volumique. Idéalement, le centre de masse de la fusée devrait se situer au milieu de celle-ci.

Il ne reste plus qu'à fabriquer des ailerons adaptés à votre fusée, en n'oubliant pas d'ajouter leur masse dans vos calculs.

Pour les ailerons et la coiffe, nous avons opté pour du PLA blanc, ce qui nous permet de réaliser ces pièces sur mesure à moindre coût grâce à notre imprimante 3D au local. Le choix du blanc est important pour éviter l'absorption des rayonnements solaires, ce qui limite la dissipation de chaleur dans le corps de la fusée lorsqu'elle est sur le pas de tir.

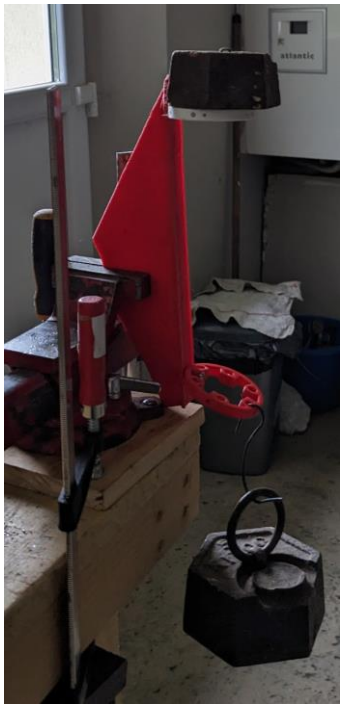


CAO de la coiffe, tube pitot, support PCB et caméra

Découvrant la modélisation et l'impression 3D pendant ce projet, nous avons décidé de tester la résistance de ces techniques. Tout d'abord, nous avons imprimé les ailerons dans le sens du bord d'attaque pour assurer une meilleure résistance. Notamment pour le test VL3 du cahier des charges pour mini-fusées, où il faut faire basculer la fusée en la tenant par l'extrémité de deux ailerons. Ce test maison s'est révélé concluant, car un aileron pouvait résister à 30 kg avant de rompre à 35 kg.

Test maison pour l'aileron avec 30kg

Nous avons également réalisé ce test pour les bagues qui positionnent les ailerons (rouge et blanche sur la photo ci-dessus). Ces bagues ont cédé à 5 kg, ce qui était dû au sens d'impression, qui n'était pas idéal et favorisait les fissures du PLA.



Test maison pour l'aileron

III) Electronique

L'électronique de la fusée est composée de deux parties. Une première partie séquenceur et l'autre expérience. Ce sont deux cartes électroniques séparées et complètement indépendantes. Elles n'ont pas la même alimentation et ne communiquent pas entre elles.

Partie séquenceur :

Hardware :

Le séquenceur de la fusée est un timer qui permet d'ouvrir la trappe du parachute au moment voulu.

Pour faire cela, nous avons utilisé une ATtiny85 (Figure 1). C'est un tout petit microcontrôleur qui nous a permis un gain de place considérable mais qui nous a aussi apporté beaucoup d'ennui.

Figure 1 Photo d'une ATtiny

Ce microcontrôleur s'alimente en 5V il a donc fallu faire un régulateur de tension. Pour cela nous avons utilisé un régulateur L7805CV (Figure 2) et trois condensateurs, un de 10µF en entrée et deux autres de 100µF et 0,1µF en sortie. Nous ajoutons aussi une diode pour protéger notre circuit. Nous pouvons donc alimenter notre ATtiny avec une pile de 9V dont la tension sera abaissée grâce à ce circuit.

Figure 2 Photo d'un régulateur de tension L7805CV

Grâce à l'ATtiny, nous allons pouvoir ouvrir notre trappe parachute mais pour cela nous avons eu besoin d'un servo moteur (Figure 3). Celui-ci s'alimente aussi en 5V, nous l'avons donc branché en sortie de notre régulateur de tension.

Figure 3 Photo d'un servo moteur

Pour faire tourner le servo moteur l'ATtiny lui envoie un signal PWM. Ce dernier va donc tourner de l'angle qui lui est demandé.

Pour détecter le décollage, nous avons ajouté à notre circuit un port jack (Figure 4). Lorsque la fusée décolle, le port jack s'arrache et la boucle qu'il formait n'est donc plus fermée, l'ATtiny détecte alors le décollage.

Figure 4 Photo d'un port jack

Et enfin pour connaître l'état de la fusée nous avons ajouté une LED bleue (Figure 5). Celle-ci doit clignoter pour nous informer lorsque le port jack est débranché. Mais lorsque ce dernier est branché, elle s'allume. Au moment où le décollage est détecté, la LED se met de nouveau à clignoter mais plus rapidement cette fois-ci.

Figure 5 Photo d'une LED bleue

Pour tester notre programme et nos branchements, nous avons fait du prototypage. C'est-à-dire, nous avons réalisé le montage sur breadboard comme sur le schéma ci-dessous (Figure 6).

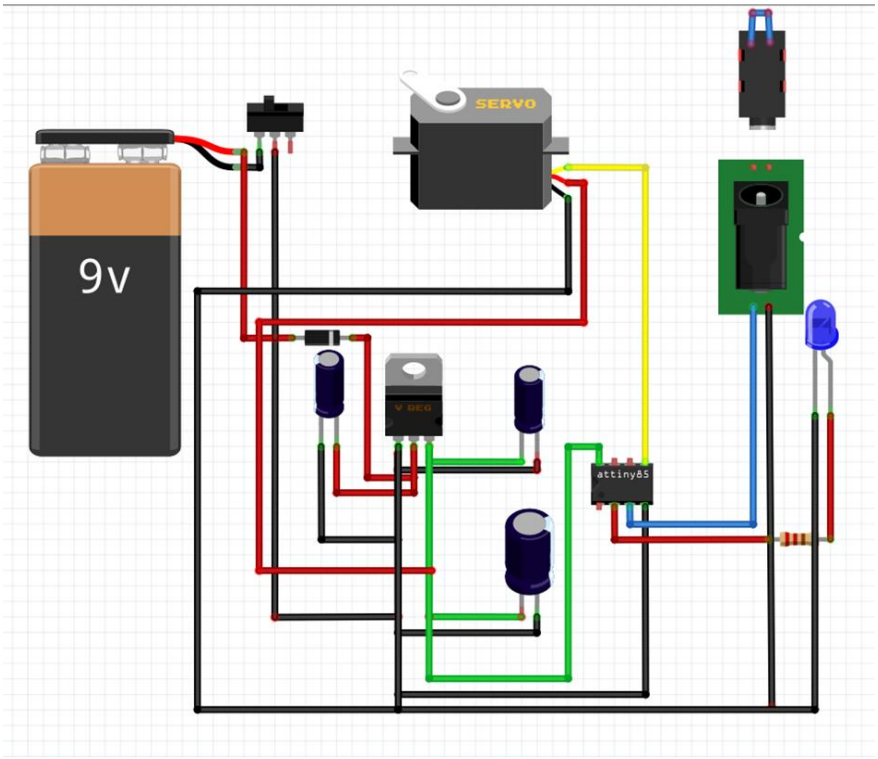


Figure 6 Schéma prototypage séquenceur

Pour embarquer tous ces composants dans la fusée, nous avons dû créer un PCB grâce au logiciel Kicad. Pour cela il faut reproduire le schéma de prototypage sur Kicad (Figure 7).

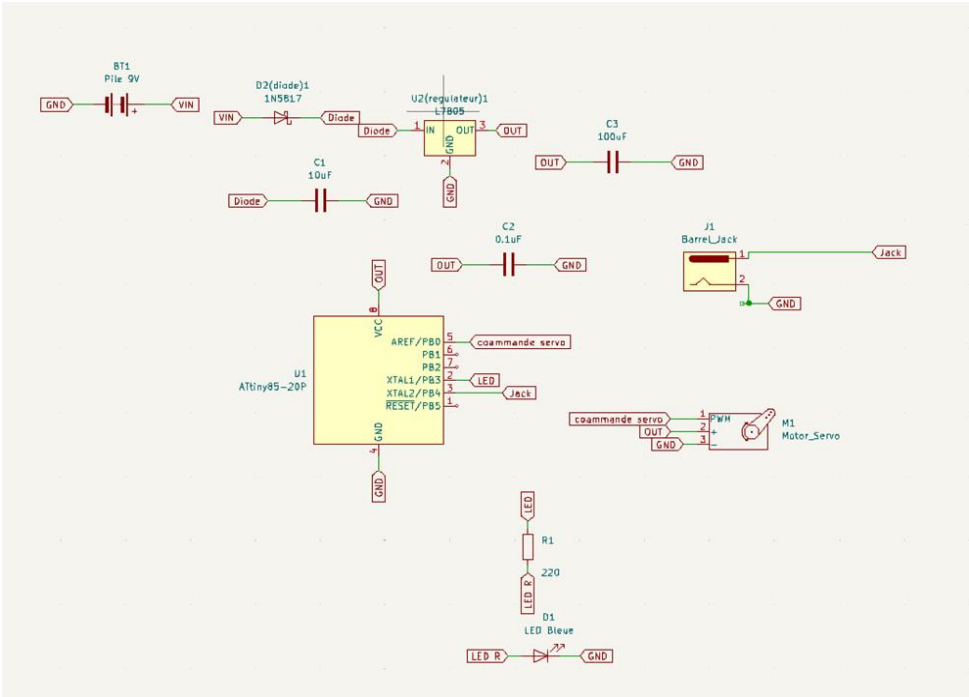


Figure 7 Schéma PCB séquenceur

Une fois cela fait, il faut attirer des empreintes à chaque composant et générer le PCB (Figure 8).

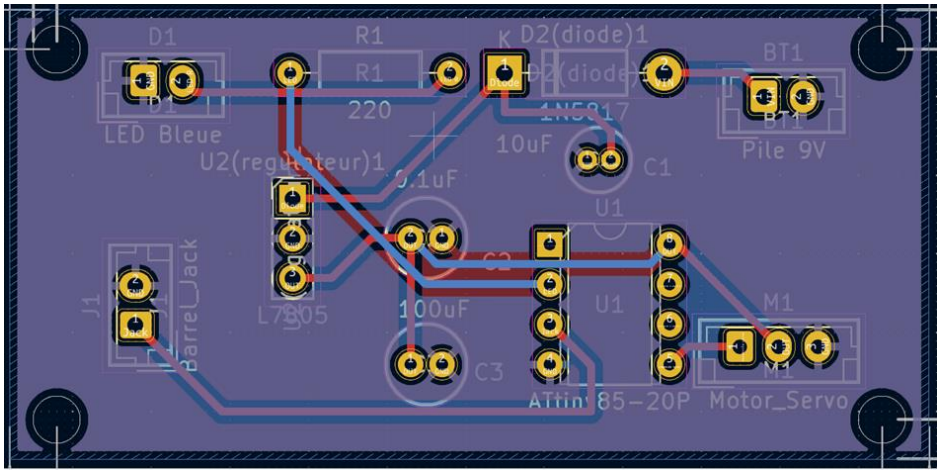


Figure 8 PCB séquenceur

Toute la partie hardware du séquenceur est désormais achevée, intéressons à la partie software.

Software :

Dans cette partie, nous allons nous intéresser à la programmation de notre ATtiny pour qu'elle ouvre la trappe au moment voulu.

Tout d'abord, nous définissons les variables dont nous allons avoir besoin dans notre programme (Figure 9).
Nous nécessitons deux variables de temps l'une qui s'initialise au lancement du programme et l'autre au décollage.

```
#include <SoftwareServo.h>

  // Définition des pins:
#define LEDbleue 3 //Relie le pin 3 à la LED bleue qui indique si la fusée est en vol
#define PortJack 4 //Relie le pin 4 au signal provenant du port jack

SoftwareServo myservo;

unsigned long GNDtime; //Déclare la variable GNDtime
unsigned long temps; //Déclare la variable temps
```

Figure 9 Définition des variables

Ensuite, nous devons définir les pins. Nous avons aussi ajouté une condition, la LED bleue doit clignoter lorsque le port jack est débranché. Tant que ce dernier ne sera pas branché, le programme sera bloqué, la suite ne s'exécutera pas tant que la condition n'est pas remplie (Figure 10).

```
void setup()
{
  // Définition des types de pins:
pinMode(LEDbleue, OUTPUT); //Définit le pin 3 comme signal sortant
pinMode(PortJack, INPUT_PULLUP); //Définit le pin 4 comme signal entrant
myservo.attach(0);              //Relie le pin 0 au servo moteur qui ouvre la trappe


while (digitalRead (PortJack) == HIGH) //Tant que le port jack est débranché, la LED va clignoter
{
clignotement(LEDbleue,500,500);//La LED bleue clignotte pour dire que le circuit est alimenté
}
}
```

Figure 10 Définition des pins et ajout d'une condition

Il ne nous reste plus qu'à définir les conditions pour que la fusée soit considérée par l'Attiny comme en vol (Figure 11).

```
void loop()
{
  if(digitalRead(PortJack) == LOW)  //Quand le port jack est branché
  {
      auSol(); //La fusée est au sol
      GNDtime = millis();
  }
  else  {

      enVole(); //Sinon elle est en vole
   }
}
```

Figure 11 Condition de vol

Notre programme fait appel à plusieurs fonctions. Il nous faut donc les créer.
Dans les boucles au sol et en vol, nous devons y écrire ce que nous souhaitons que le programme fasse lorsque la fonction est appelée (Figure 12).

```
void auSol() //Quand le port jack est branché la fusée est au sol
{
  digitalWrite(LEDbleue, HIGH);//La LED bleue s'allume
     myservo.write(1);             // Définit le servo moteur comme dans sa position initiale
  SoftwareServo::refresh();       // rafraichit toutes les 50ms ou garde le servo moteur à jour

  //Pas d'envoie dans la boucle d'ouverture de la trappe donc le servo moteur ne reçoit pas de signal la trappe du parachute reste fermée
}

void enVole() //Quand le port jack est débranché la fusée est en vole
{
  temps = millis(); //A cet instant le temps vaut 0, on l'initialise
  clignotement(LEDbleue,1000,1000); // La LED bleue clignotte plus lentement
  if (temps - GNDtime >= 6000) //Au bout de 6 secondes le signal est envoyé au servo moteur !!! VALEUR A MODIFER SI LE STABTRAJ CHANGE
  {clignotement(LEDbleue,800,800); //La LED bleue clignotte
    ouverture_trappe();//Envoie dans la boucle pour ouvrir la trappe, le servo moteur va recevoir un signal, la trappe va s'ouvrir et le parachute sera ejecté
  }
}
```

Figure 12 Boucles au sol et en vol

Pour la boucle clignotement (Figure 13), Nous reprenons les valeurs données précédemment et ainsi dès que la boucle est appelée elle a directement les valeurs à utiliser.

```
void clignotement(int pin, int on, int off) //Reprend les valeurs intiales pin=LEDbleue, on=500ms et off=500ms
{
  if (millis()%(on+off)<on) //Si la division par (on+off) est inférieure à on
  digitalWrite(pin, HIGH); //Alumme la LED
  else
  digitalWrite(pin, LOW); //Eteint la LED
}
```

Figure 13 Boucle clignotement

La dernière boucle appelée est celle de l'ouverture trappe (Figure 14), c'est elle qui lorsqu'elle est appelée donne l'ordre au servo de tourner.

```
void ouverture_trappe()             //Boucle qui permet l'ouverture de la trappe et l'éjection du parachute
{
    myservo.write(89);              // Dit au servo moteur de tourner de 89°
    SoftwareServo::refresh();        // Garde le servo moteur à jour
}
```

Figure 14 Boucle ouverture trappe

La partie séquenceur est terminée, passons à la partie expérience.

Partie expérience :

Hardware :

Notre expérience est un tube Pitot auquel est raccordé un capteur de pression différentielle. Grâce à ce capteur nous pourrons obtenir la pression dans laquelle la fusée évolue mais aussi sa vitesse.

Pour ce faire nous nécessitons tout d'abord un tube Pitot (Figure 15). Celui-ci est composé de deux tubes internes, l'un pour la prise statique et l'autre pour la prise dynamique.

Figure 15 Photo d'un tube Pitot

Ensuite nous avons choisi comme capteur de pression le MXP 5010 (Figure 16) car il a une plage de mesure qui correspond aux valeurs que nous allons mesurer. La prise de droite du capteur est la prise dynamique elle sera donc reliée à la prise dynamique du Pitot et il en sera de même pour la prise statique.

Figure 16 Photo capteur de pression

Maintenant il faut que l'on soit capable de lire les valeurs que le capteur renvoie et de les utiliser pour obtenir la pression et la vitesse.

Pour cela nous allons utiliser une carte Arduino nano (Figure 17) celle-ci sera capable de lire les valeurs analogiques envoyées par le capteur et de les transformer en une pression et une vitesse. Le capteur de pression sera branché sur un pin analogique.

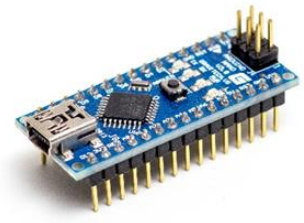


Figure 17 Photo d'une carte Arduino nano

Pour savoir si notre capteur est alimenté et renvoie des valeurs cohérentes, nous avons ajouté deux LED, une verte et une rouge (Figure 18) qui seront reliées à deux pins digitaux de l'Arduino nano.

Figure 18 Photo LED rouge et verte

Et enfin pour enregistrer les données que le capteur nous renvoie, nous avons besoin d'un module et d'une carte SD (Figure 19).

Figure 19 Photo module SD et carte SD

Pour tester notre programme et nos branchements, nous avons fait du prototypage. C'est-à-dire, nous avons réalisé le montage sur breadboard comme sur le schéma ci-dessous (Figure 20).

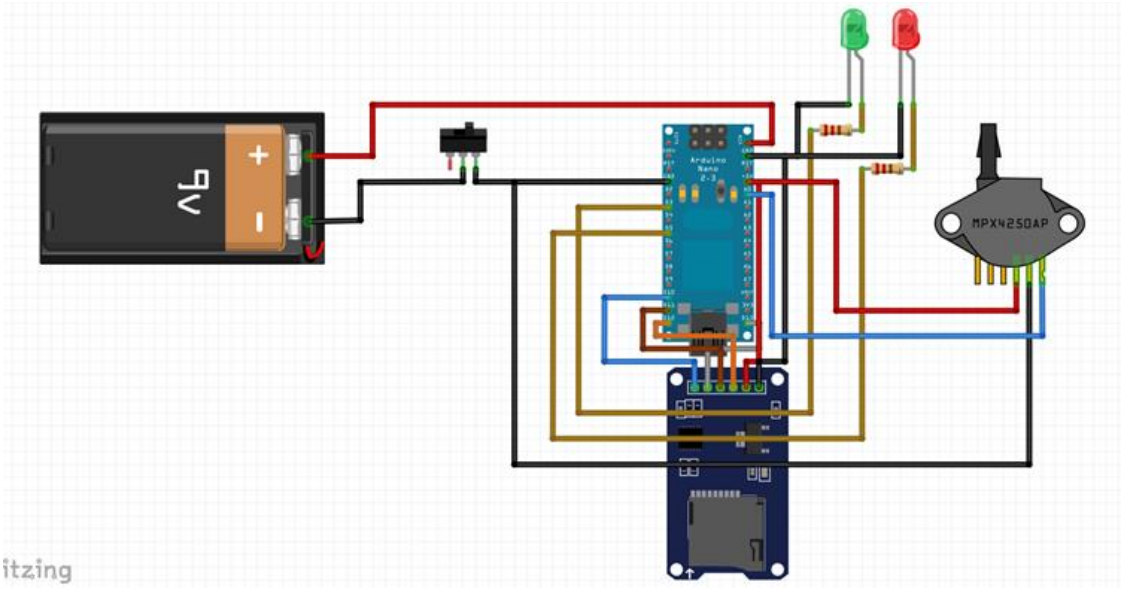


Figure 20 Schéma prototypage expérience

Pour embarquer tous ces composants dans la fusée, nous avons dû créer un PCB grâce au logiciel Kicad. Pour cela il faut reproduire le schéma de prototypage sur Kicad (Figure 21).

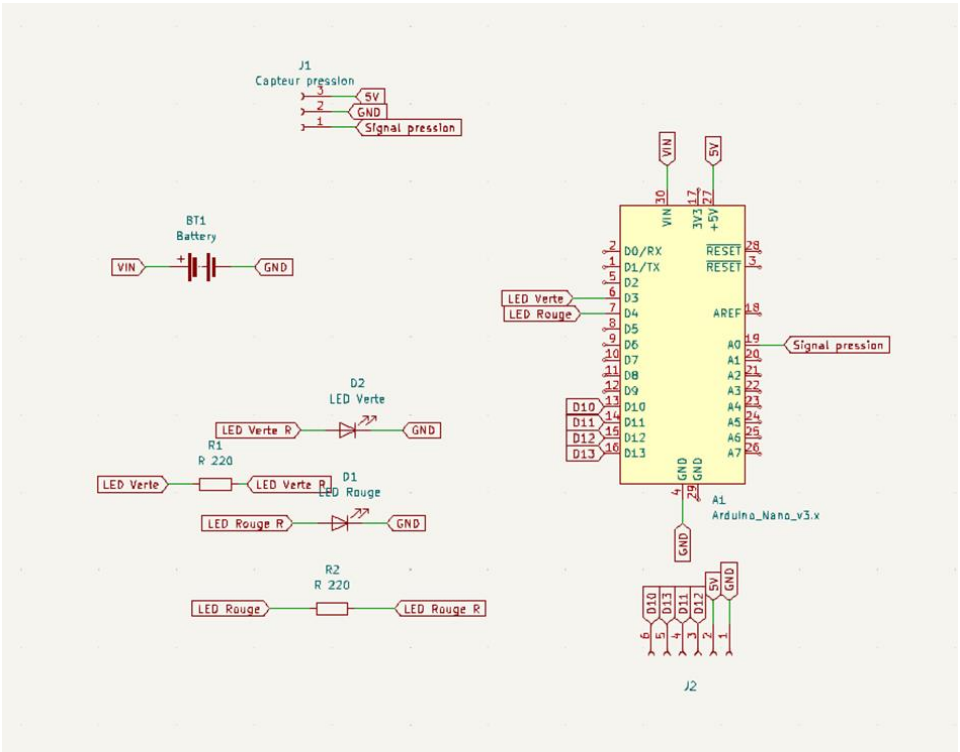


Figure 21 Schéma PCB expérience

Une fois cela fait, il faut attirer des empreintes à chaque composant et générer le PCB (Figure 22).

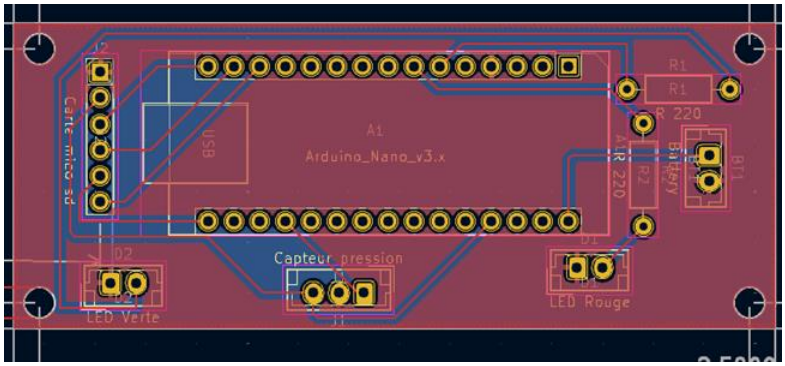


Figure 22 PCB expérience

Toute la partie hardware de l'expérience est désormais achevée, intéressons-nous à la partie software.

Software :

Dans cette partie nous allons nous intéresser à la programmation de la carte Arduino pour qu'elle enregistre sur la carte SD les valeurs de pression et de vitesse calculées.

Tout d'abord définissons les pins et déclarons les variables dont nous allons avoir besoin (Figure 23).

```
float tension, pression, vitesse; // Tension, pression et vitesse
float somme, pression_offset; // Somme des 10 valeurs de tension, moyenne de l'offset de pression
int valeur_capteur; // Valeur analogique du capteur

//Définition des pins
const int pin_capteur = A0; // Pin A0 ==> Capteur | Plage 0-1023
#define LEDverte 3 // Pin 3 ==> LED verte | Différence de pression présente
#define LEDrouge 4// Pin 4 ==> LED rouge  | Différence de pression nulle/trop faible
```

Figure 23 Déclaration des variables et définition des pins

Ensuite définissons les pins que nous utilisons (Figure 24).

```
void setup()
{

  Serial.begin(9600);

 // Définition des types de pin
 pinMode(LEDverte, OUTPUT); // LED verte ==> Sortie (digital)
 pinMode(LEDrouge, OUTPUT); // LED rouge ==> Sortie (digital)
 pinMode(pin_capteur, INPUT); // Capteur ==> Entrée (analog)
 pinMode(13, OUTPUT); // Pin LED interne ==> Sortie (digital)
```

Figure 24 Définition des pins

Pour obtenir des valeurs cohérentes, nous devons ajuster un offset avant le début des calculs fait avec les valeurs que renvoie le capteur (Figure 25, Figure 26). Les calculs de tension et pression sont faits dans les fonctions lecture tension et calcul pression.

```
//---------- Calcul offset ----------

Serial.println("Calibration de l'offset...");

for (int i = 0; i < 10; i++)
{ // Boucle de 10 calculs de la pression
  somme += calculPression(); // Somme accumulant la pression calculée à chaque itération
  digitalRead(13) == LOW ? digitalWrite(13, HIGH) : digitalWrite(13, LOW); // Clignotement de la LED interne ("L" sur la carte)
  /* Syntaxe du "if()" raccourci :
   *  condition à vérifier ? faire ceci : sinon faire cela
   *  En version classique ça donne :
   *  if(digitalRead(13) == LOW){digitalWrite(13, HIGH);} else {digitalWrite(13, LOW);}
   */
  delay(500); // Délai entre chaque lecture
}

pression_offset = somme/10; // L'offset est la moyenne des 10 lectures
Serial.print("Offset de la pression : ");
Serial.print(pression_offset, 3); // Affichage de l'offset à 3 chiffres après la virgule
Serial.println(" Pa");
Serial.println("... C'est prêt!");
Serial.println(" ");

delay(1000); // Délai 1s
```

Figure 25 Offset

```
//----------- Lecture & calculs des données -----------

tension = lectureTension(); // Tension (V)
pression = calculPression() - pression_offset; // Pression calibrée (Pa)
```

Figure 26 Lecture des données avec l'offset

Une fois la pression obtenue, nous pouvons obtenir la vitesse grâce au théorème de Bernoulli (Figure 27).

```
pression = (pression < 0 ) ? 0 : pression; // Si la pression est négative, la maintenir à 0 Pa
vitesse = sqrt(2*pression / 1.225); // Calcul de la vitesse (m/s), théorème de Bernoulli
```

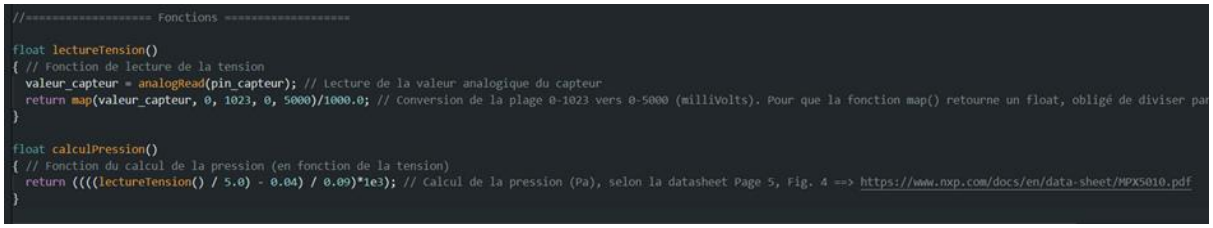
Figure 27 Calcul de la vitesse

Maintenant, programmons la carte pour que les LED s'allument suivant les valeurs que renvoie le capteur (Figure 28). Si le capteur renvoie une valeur analogique inférieure à 30 ou supérieur à 923, la LED rouge s'allume sinon c'est la LED verte.

```
//----------- Etat des LEDs -----------

if (valeur_capteur <= 30 || valeur_capteur >= 923)
{
    digitalWrite(LEDrouge, HIGH); // Sinon la LED rouge s'allume
    digitalWrite(LEDverte, LOW); // Et la LED verte s'éteint
}
else
{
    digitalWrite(LEDverte, HIGH); // La LED verte s'allume
    digitalWrite(LEDrouge, LOW); // La LED rouge s'éteint
}
```

Figure 28 Etat des LED

Intéressons-nous plus précisément aux fonctions qui nous permettent de lire la tension et de calculer la pression (Figure 29).



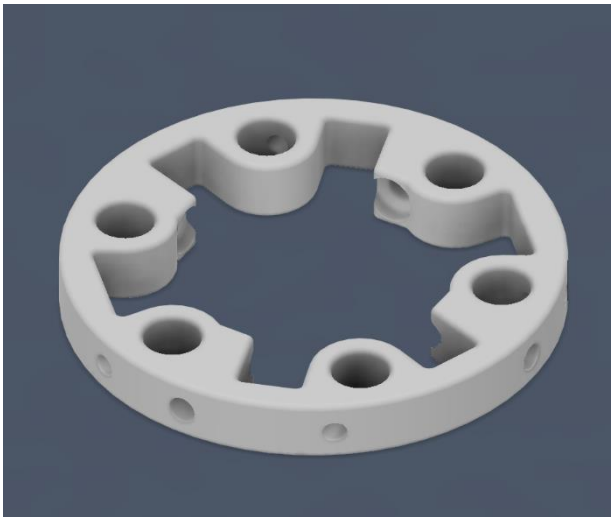
```
//================== Fonctions ==================

float lectureTension()
{ // Fonction de lecture de la tension
  valeur_capteur = analogRead(pin_capteur); // Lecture de la valeur analogique du capteur
  return map(valeur_capteur, 0, 1023, 0, 5000)/1000.0; // Conversion de la plage 0-1023 vers 0-5000 (milliVolts). Pour que la fonction map() retourne un float, obligé de diviser par
}

float calculPression()
{ // Fonction du calcul de la pression (en fonction de la tension)
  return ((((lectureTension() / 5.0) - 0.04) / 0.09)*1e3); // Calcul de la pression (Pa), selon la datasheet Page 5, Fig. 4 ==> https://www.nxp.com/docs/en/data-sheet/MPX5010.pdf
}
```

Figure 29 Fonctions

La partie expérience est terminée. Nous ne traiterons pas la partie enregistrement sur la carte SD car celle-ci n'a pas fonctionnée lors du vol à cause d'un problème de programmation.

IV) Assemblage et intégration

Pour cette partie, nous avons dû réfléchir à la manière de fixer les canettes sous les ailerons. Nous avons donc positionné les canettes et ensuite vissé les ailerons de l'intérieur. Bien sûr, l'opération était délicate car nous avons dû mettre nos mains entre les canettes coupantes pour visser les ailerons à l'aide d'une bague en PLA, afin que les ailerons soient perpendiculaires à la fusée.



Bague pour fixer les ailerons

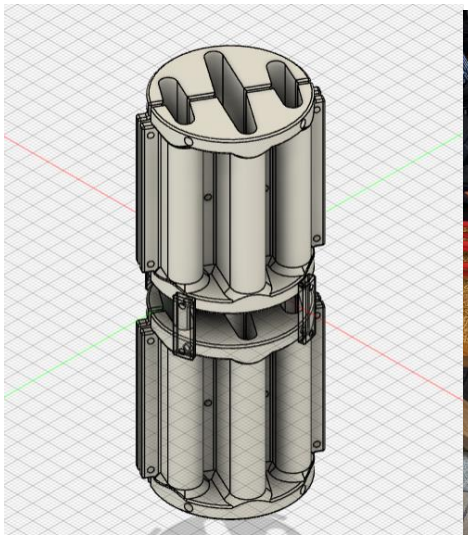
Ensuite nous avons pu visser les canettes aux bagues en bois.

Rendu final des ailerons

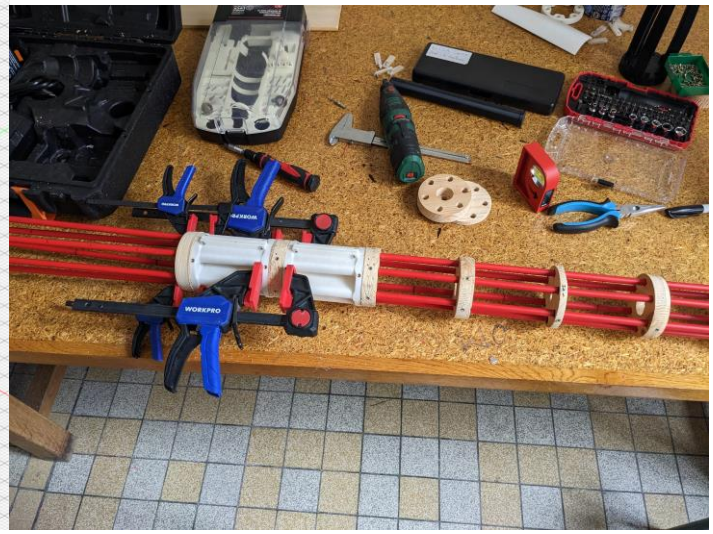
La seconde étape a été de monter toutes les bagues en bois que nous avons découpées dans du bois de sapin, et pour percer les trous nous avons utilisé une perceuse a colonne et un gabarit en 3D, qui a était plus que nécessaire pour être droit et que la fusée soit droite par la suite !

Rendu final de l'assemblage.

Pour placer les bagues en bois à la bonne distance, nous nous sommes aidés d'un support, qui nous permettait dans un premier temps de fixer les éléments fixes (tiges et bagues en bois).

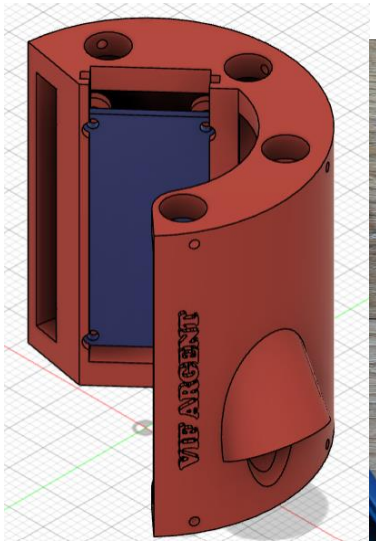


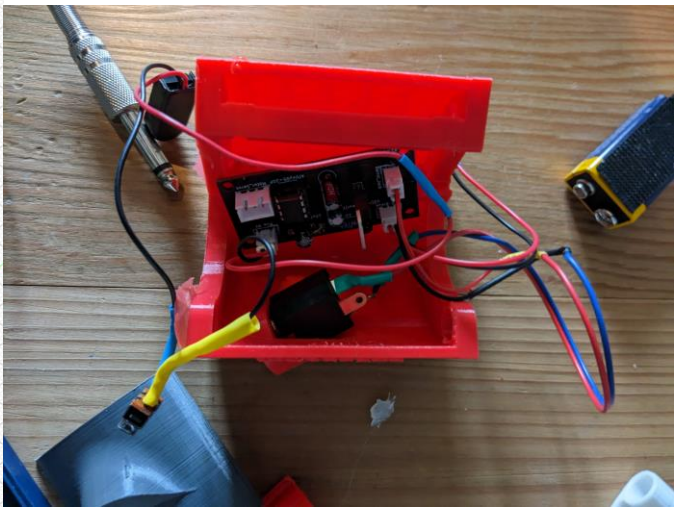
Rendu des supports pour aider au montage



Montage et fixation des bagues et des tiges

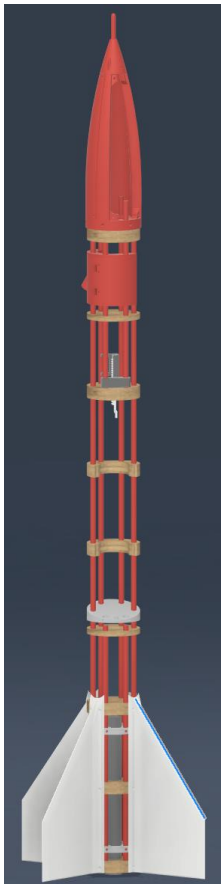
L'électronique du séquenceur fut logée dans son compartiment 3D, qui accueille la prise jack, les interrupteurs et les LED, ainsi que le PCB du séquenceur.





Boitier pour le PCB, la prise Jack, les interrupteurs et les LED.

En réalité, cette partie était initialement en 3 pièces 3D distinctes, mais pour un meilleur accès, nous avons décidé de la combiner en une seule. Le point négatif est que dans une mini-fusée vous n'avez pas beaucoup de place.



CAO

Servo et système de récupération :

Cette phase du projet s'est avérée particulièrement complexe, car elle a exigé de trouver une solution pour concevoir un système linéaire qui puisse non seulement s'intégrer parfaitement dans la fusée, mais aussi s'accrocher solidement aux tiges de fixation. Cette contrainte de design nous a poussés à faire preuve d'ingéniosité, d'autant plus que l'espace disponible à l'intérieur de la fusée était extrêmement limité.

Intégration du Servo et de la pile

Afin de répondre aux besoins en alimentation du servo-moteur, nous avons opté pour l'utilisation d'une pile. Cependant, ce choix a introduit un nouveau défi : celui de l'intégration de la pile dans l'espace restreint. Compte tenu de ces contraintes, nous avons décidé de fixer la pile à l'aide d'un Velcro, ce qui nous a permis de maintenir une certaine flexibilité dans l'agencement interne tout en assurant la stabilité du composant.

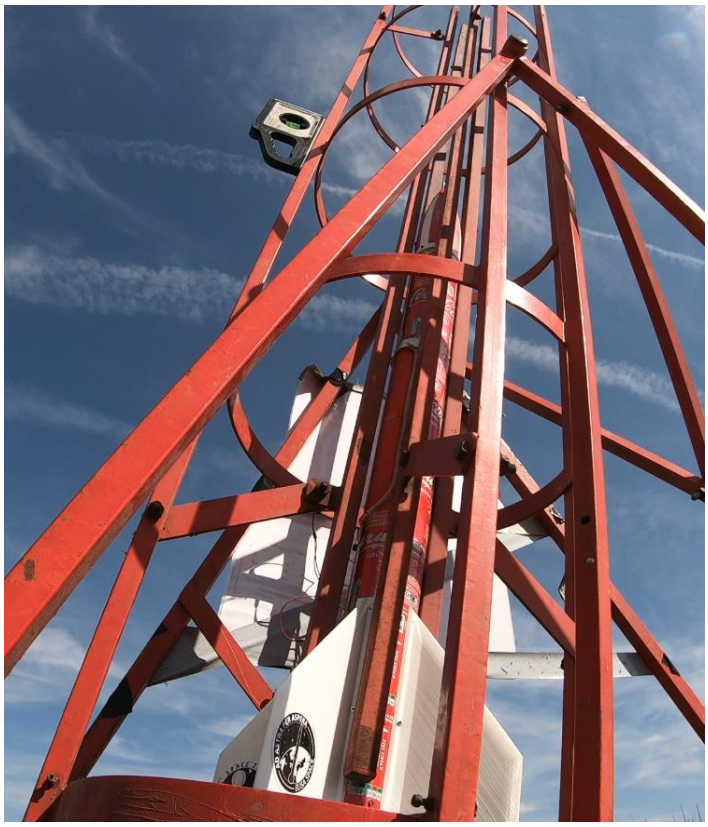
Malheureusement, le choix d'utiliser deux pièces en PLA pour la structure s'est révélé problématique…

V) C’Space 2024

Vol de Vif Argent :

La semaine tant attendue du lancement est enfin arrivée. Nous étions prêts, nous avons passé les contrôles avec succès, et une fois qualifiés, nous avons lancé directement dès le deuxième jour. Après les contrôles, impossible de toucher à la fusée, alors l’attente toute la matinée avant le lancement a été longue, et la nuit précédant le vol, très stressante…

Prêts pour le vol, nous nous sommes dirigés vers la zone de lancement (ZAS). Après une attente interminable, nous avons finalement procédé au lancement. Le décollage a été magnifique, un vol balistique parfait et stable… mais malheureusement, notre système de récupération n'a pas fonctionné.



Vif Argent en rampe

Nous pensons que le problème vient de la chaleur, car deux pièces en PLA faisaient partie du système de récupération. Il est probable que ces pièces se soient dilatées sous l'effet de la chaleur, empêchant le système de se déclencher correctement. Le servo, lui, a bien fonctionné – on l’entend distinctement sur la vidéo.

Nous n'avons malheureusement pu récupérer aucune des données de l'expérience à cause du vol balistique. La principale raison de cet échec est liée à la configuration de l'écriture des données sur la carte SD. En effet, celle-ci n'était pas découpée en paquets de données, ce qui a conduit à une écriture unique et continue de 20 minutes seulement, rendant impossible la récupération des informations après le vol.

Pour améliorer la collecte de données lors de futurs lancements, plusieurs points doivent être pris en compte. Tout d'abord, il serait essentiel de fragmenter les fichiers de données, permettant ainsi de sauvegarder des informations à différents moments du vol, même si celui-ci est balistique. Ensuite, la durée de l'écriture sur la carte SD devrait être étendue bien au-delà de 20 minutes. Cela garantirait que les données soient capturées même si la fusée reste en attente prolongée sur la rampe avant le décollage.

Ces améliorations seront cruciales pour maximiser les chances de succès lors des prochains essais et assurer une collecte de données plus fiable et complète, indépendamment des aléas du lancement.

C'est une leçon précieuse pour nous. Il est possible que nous devions revoir les matériaux utilisés pour ces pièces critiques, ou ajuster la conception afin d'éviter que la chaleur n'affecte le fonctionnement du système. Cette expérience nous permettra de perfectionner de nouveau projet pour les prochains C’Space.

Vif Argent au décollage

Travail en équipe :

Au début de l’année, l'excitation autour de la construction d'une fusée a rassemblé un petit groupe enthousiaste autour du projet "Vif Argent". Au départ, nous étions six et nous organisions de petites réunions informelles soit chez l’un soit chez l’autre, ainsi que des séances les jeudis après-midi qui nous permettaient de répartir les tâches, de visualiser les objectifs et de prendre en compte les contraintes du projet. Cependant, ces réunions à domicile présentaient un inconvénient : les discussions dérivaient souvent hors du cadre du projet, ce qui nuisait à notre efficacité.

De plus, aucun d'entre nous ne possédait réellement de compétences en programmation, modélisation, conception ou gestion de projet. Nous nous sommes donc répartis les tâches en fonction des intérêts de chacun : Lana et Hasna se sont chargées de l'électronique et de la programmation, Carl et Benjamin de la conception et de la modélisation, tandis que Lucien et Thibault se sont concentrés sur la construction, le choix des composants et l'électronique de la caméra. Cependant, cette répartition du travail n’était pas idéale en termes de charge de travail, car nous ne savions pas combien de temps nécessiteraient le développement de chaque partie et l’apprentissage des compétences nécessaires. Rapidement, certains membres se sont retrouvés à en faire plus que d’autres, ce qui a généré des tensions au sein de l’équipe.

Equipe du projet Vif Argent

Idéalement, dès que des tensions apparaissent, il est important d’organiser une petite réunion dans un cadre calme, par exemple dans un appartement, pour redistribuer les tâches de manière équitable, réorganiser le projet autour de deadlines, et raviver la motivation de ceux qui la perdent en cours de route. Une fois cela fait, profitez de l'occasion pour vous détendre ensemble ! Souvent, c’est à travers ces projets que naissent de véritables amitiés.

Malheureusement, nous aurions dû appliquer ces conseils bien plus tôt. Hasna a quitté l’école et a donc abandonné le projet peu après son début. Thibault et Carl ont, eux aussi, commencé à perdre leur motivation. Et lorsque l’association nous a informés, vers le mois de mai, qu'elle ne pourrait pas financer le voyage au C'Space pour toute l’équipe restante, cela a donné le coup de grâce à leur engagement.

Finalement, la fusée a été lancée, mais le fait que certaines personnes n'aient pas pu aller jusqu'au bout est aussi décevant qu'un vol balistique.

VI) Remerciement

Nous tenons à exprimer notre profonde gratitude à tous nos partenaires, sans qui ce projet n'aurait jamais pu voir le jour. Leur soutien indéfectible, qu'il s'agisse de financements, de fournitures de matériaux ou de conseils techniques, a été essentiel pour surmonter les nombreux défis rencontrés tout au long de cette aventure. Grâce à leur engagement et à leur confiance en notre vision, nous avons pu concrétiser ce projet ambitieux. Nous espérons que cette collaboration fructueuse se poursuivra dans le futur, et nous sommes reconnaissants de leur contribution précieuse qui a rendu ce rêve possible.