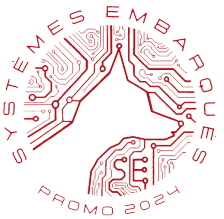


Rapport de Projet : Semestre 7

Sujet : Robot ramasseur de déchet

Présenté par : DEFFRENNES Vianney - DELANNOY Jason - THOMAS Gabriel

Sommaire :

Introduction

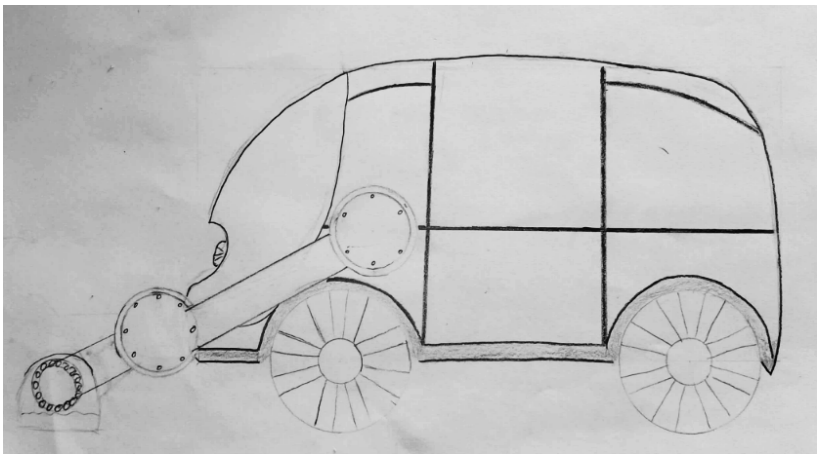
Ce dossier est le rapport de projet du groupe de Vianney Deffrennes, Jason Delannoy et Gabriel Thomas qui correspond au sujet 7. Il intervient dans le module “Projet” du semestre 6 de Systèmes Embarquées et se poursuivra au semestre 7 et 8.

Ce rapport de projet fait suite à celui envoyé lors du Semestre 6. Dans ce dossier, nous évoquerons les changements de plans proposés pour le robot ainsi que notre avancée à la fin du semestre avec notamment la répartition des tâches et nous finirons par parler des tâches à finir pour l'année prochaine.

I. Résumé du Semestre 6 de Trashy et nouvelle piste

Lors du semestre 6, nous avions établi les études de fonctionnement et de marché de notre projet. Notre projet consiste à concevoir et à mettre en place un système mobile autonome (volant ou roulant) permettant la collecte de déchets sur le campus grâce à de la reconnaissance d’image.

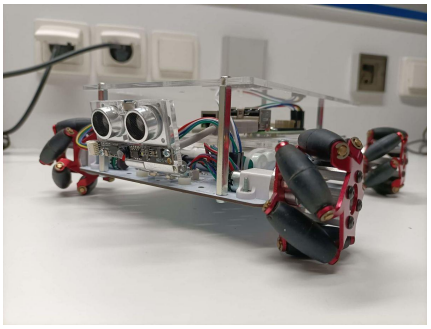
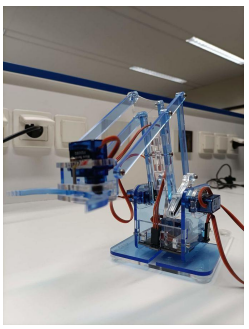
Nous avions convenu d’un robot roulant nommé “Trashy” d’une longueur d’environ 50cm et dont le design avait été imaginé ci-dessous :



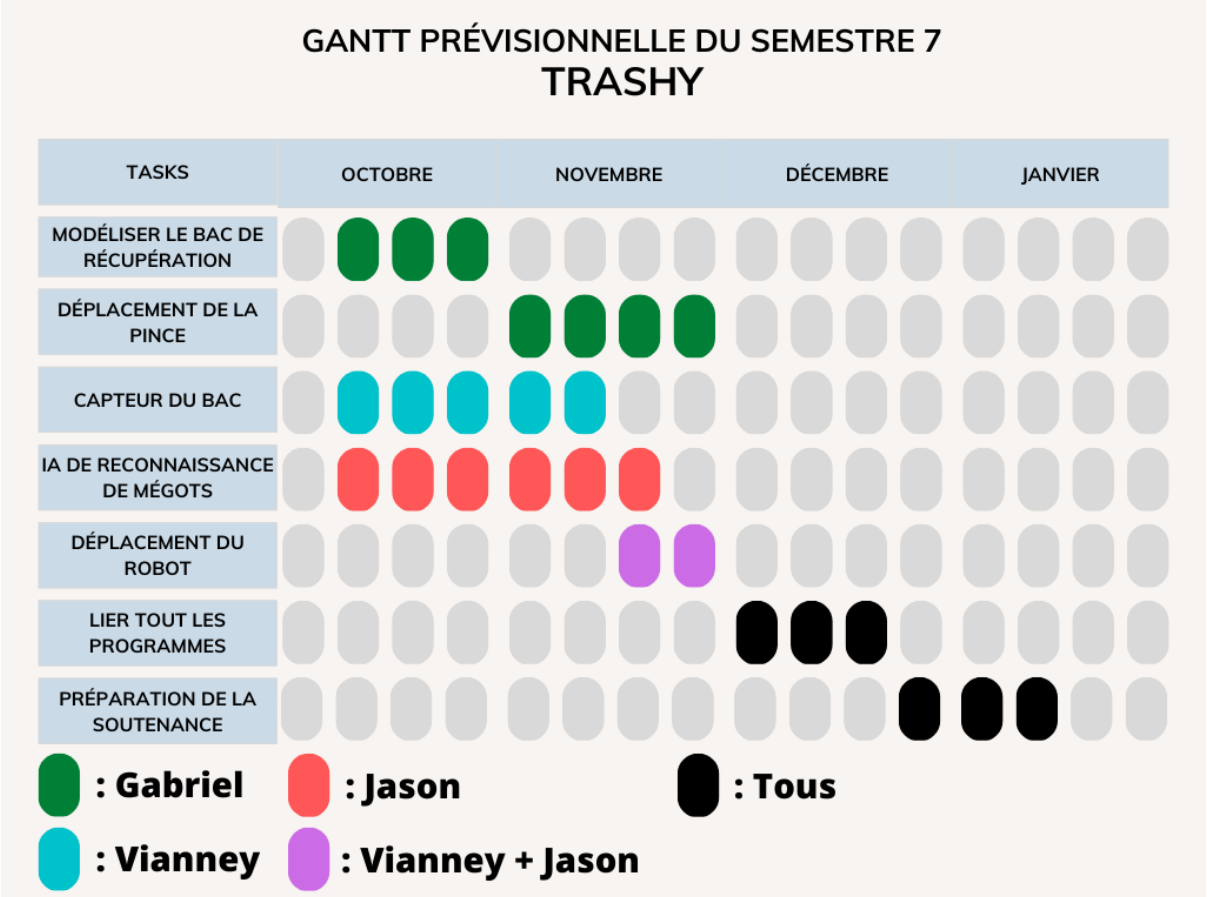
Cependant, et après discussion avec notre tuteur de projet, nous avons convenu à un robot ramasseur de mégots de cigarette seulement.

II. Avancement du projet

Nous avons reçu une base mobile programmable ainsi qu'une pince que nous allons utiliser pour la conception du prototype de Trashy :

Nous avons aussi réparti les tâches entre les membres du groupe et établi un diagramme de Gantt prévisionnelle pour le semestre 7 :



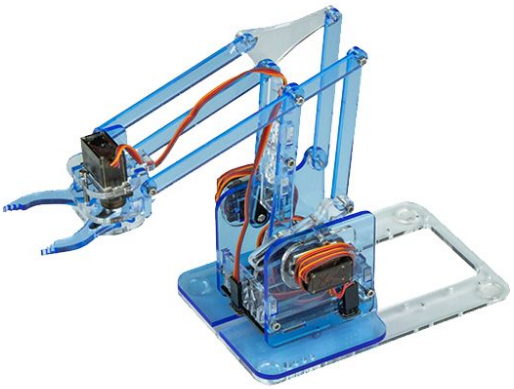
Nous allons donc dans la suite de ce dossier, vous présenter l'avancement dans les différentes parties du projet.

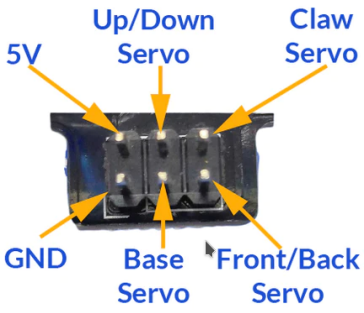
A. Contrôle de la pince

Premièrement, nous avons utilisé la pince MeArm 1.0 que nous avons relié à une carte Arduino

Pour que le robot puisse être capable de prendre les mégots, nous avons dû coder un algorithme en Arduino pour que la pince puisse avancer d'avant en arrière, tourner sur sa base, ouvrir et fermer la pince. Nous avons utilisé la librairie arduino servo.h pour nous aider. Nous avons également trouvé les numéros des pins de la pince correspondant sur internet

Remarque : le délai de 15 ms dans les fonctions sert à contrôler plus doucement les servos moteurs et éviter de les endommager, de même que créer un tableau qui s'incrémente pour éviter les sauts de positions






```
#include <Servo.h> //Ajout de la bibliothèque servo
Servo base, claw, up_down, front_back;
int angle = 0;

void setup() {
   up_down.attach(22);
   base.attach(24);
   claw.attach(26);
   front_back.attach(28);
   Serial.begin(9600);
}
```

```
void recule(){
   for(angle = 180; angle >= 1; angle -= 1){
      front_back.write(angle);
      delay(15);
   }
   delay(100);
}

void avance(){
   for(angle = 0; angle < 180; angle += 1){
      front_back.write(angle);
      delay(15);
   }
   delay(100);
}

void bas(){
   for(angle = 180; angle >=0 ; angle -= 1){
      up_down.write(angle);
      delay(15);
   }
   delay(100);
}


void haut(){
   for(angle = 0; angle < 180; angle += 1){
      up_down.write(angle);
      delay(15);
   }
   delay(100);
}

void ferme(){
   for(angle = 0; angle < 95; angle += 1){
      claw.write(angle);
      delay(15);
   }
   delay(100);
}

void ouvrir(){
   for(angle = 95; angle >= 0; angle -= 1){
      claw.write(angle);
      delay(15);
   }
   delay(100);
}

void basedroite(){
   for(angle = 0; angle < 180; angle += 1){
      base.write(angle);
      delay(15);
```

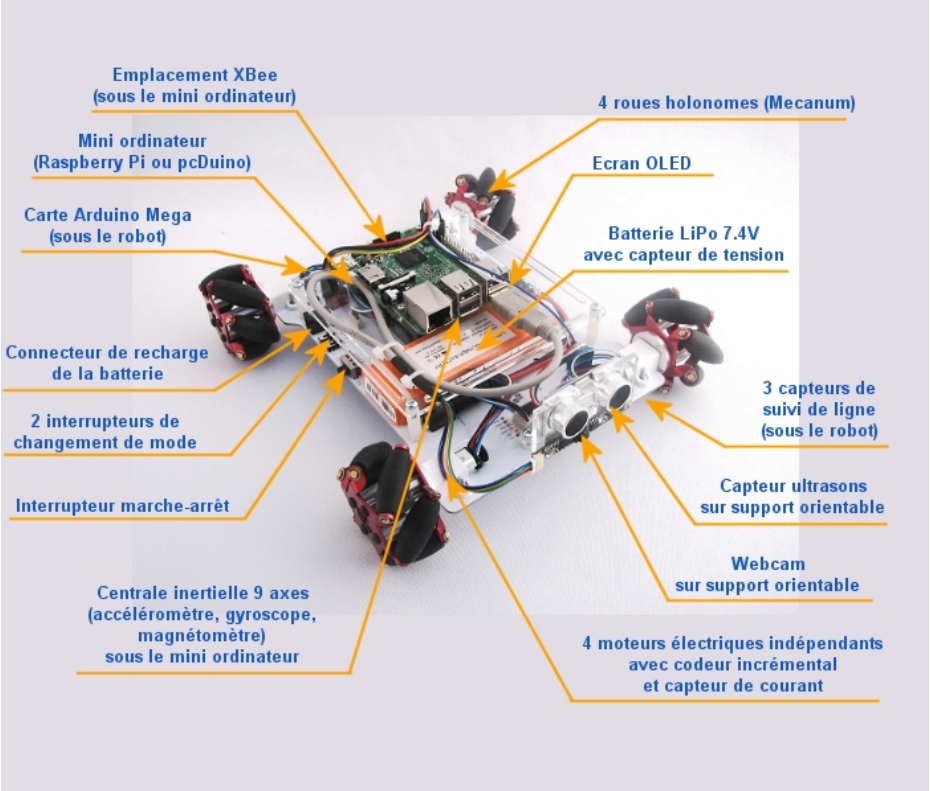
```
    }
    delay(100);
}

void basegauche(){
    for(angle = 180; angle >= 0; angle -= 1){
        base.write(angle);
        delay(15);
    }
    delay(100);
}

void loop(){
    up_down.write(0); //mise à zéro des servo moteurs
    base.write(0);
    claw.write(0);
    front_back.write(0);
    ouvrir();
    avance();
    ferme();
    recule();
    basedroite();
    ouvrir();
    ferme();
    basegauche();
}
```

B. Fonctionnement du robot

Le robot que nous avons reçu est un robot T-Quad, développé par 3sigma. T-Quad est un AGV (Automated Guided Vehicle) qui possède 4 roues motrices omnidirectionnelles chacune commandées par son propre moteur électrique avec codeur incrémental (qui nous servira de capteur de position) et capteur de courant.



Ces moteurs étaient commandés par une Raspberry Pi 3. Cependant, la notre était défectueuse: nous ne pouvions pas la connecter à internet, ce qui signifie que nous ne pouvions pas télécharger ROS, un système d'exploitation pour robots essentiel pour notre application. Nous nous sommes donc penchés vers une Raspberry pi 4.

Après avoir installé l'OS de la Raspberry et configuré internet, nous avons installé les packages ROS suivants :

ros-kinetic-usb-cam	Fournit le flux vidéo de la caméra branché en usb.
ros-kinetic-rosbridge-server	Fournit une interface de connexion à client non Ros pour la communication
ros-kinetic-rosserial-arduino	Fournit un protocole de communication ROS pour l'arduino

Nous avons téléchargé les bibliothèques et décompressé dans le répertoire des librairies de votre IDE arduino avant le téléversement du firmware arduino des T-Quads dans la carte Mega. suivantes

Nom de la librairie	Utilité	Provenance
DigitalWriteFast	Ecrire plus rapidement sur les Entrées/Sorties de l'Arduino	http://www.3sigma.fr/telechargements/digitalWriteFast.zip
EnableInterrupt	Gère les interruptions sur l'Arduino	http://www.3sigma.fr/telechargements/EnableInterrupt.zip
NewPing	gère le capteur ultrason du robot T-Quad	http://www.3sigma.fr/telechargements/NewPing.zip
roslib	permet de faire le lien entre ROS et arduino	commande terminal: rosrun rosserial arduino make_librairies.py

C. Gestion des déchets

Pour ramasser les déchets, le robot est équipé d'un bac de récupération auquel est attaché un laser et une photorésistance. A l'aide d'un algorithme, nous sommes capable de savoir si un objet est passé entre le laser et la photorésistance et d'allumer une led correspondant à une situation respective :

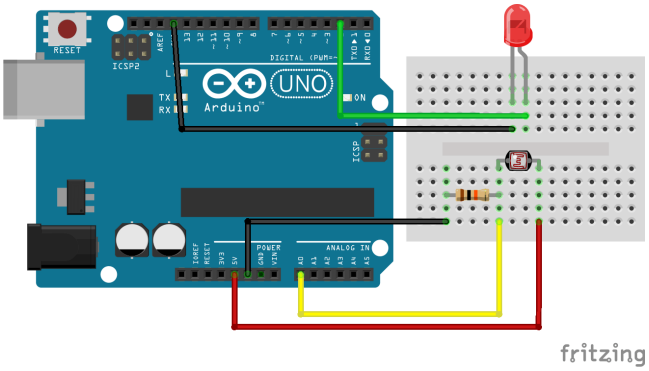


Schéma simplifié (en réalité nous avons 2 Leds et 1 laser)

1) Le bac est rempli

Nous savons si le bac est rempli lorsque le temps de coupure du faisceau laser sur la photorésistance est plus long que 1 seconde, cette action amène le robot à allumer une led 1 (bac plein)

2) Un objet est passé de façon brève entre le laser et la photorésistance

Nous savons si le robot à bien ramasser un déchet si le temps de coupure du faisceau laser est inférieur à 100ms, dans ce cas, le robot allume la led 2 (déchet bien ramassé)

Code de la gestion de déchets en arduino :

```
int lightPin = 0;
int ledPin1 = 9;
int ledPin2 = 10;
int bref =0;
int loong=0;

void setup(){
     pinMode (ledPin1, OUTPUT); //initialisation
     pinMode (ledPin2, OUTPUT);
     Serial.begin(9600);
}

void loop(){
     int seuil = 850;   //on initialise un seuil pour lequel la led
s'allumera
     bref=(analogRead (lightPin)< seuil) ;
     delay(20); //dans le cas d'une détection brève
     loong=(analogRead (lightPin)< seuil);
     if (bref==1 && loong==1){ //si la photorésistance //détecte
toujours le laser (plus de 20ms)         digitalWrite (ledPin1,
HIGH);  //on allume la led 1 pendant 1s
          delay(1000);
          }
     else if (bref==1 && loong==0) //si la photorésistance ne
//détecte plus le laser (soit moins de 20ms)
     {
          digitalWrite (ledPin2, HIGH); //on allume la led 2 pendant
1s
          delay(1000);
     }
     else{ //on éteint les leds ensuite
       digitalWrite (ledPin2, LOW);
       digitalWrite (ledPin1, LOW);
     }
 }
```

D. Développement de l'IA

Durant ce semestre, nous nous sommes concentrés sur la compréhension et la recherche de comment créer une IA capable de détecter les mégots de cigarette et comment chaque partie va communiquer entre elles. Pour réaliser cette partie, notre tuteur de projet nous as prêté une **Nvidia Jetson Nano**, un micro-ordinateur de chez Nvidia très utilisé dans le développement d'IA :

Pour développer l'IA, nous nous inspirons des vidéos expliquant le fonctionnement et la façon de faire présent sur la chaîne youtube ***Nvidia Developper*** qui est la chaîne youtube officielle des développeurs Nvidia ainsi que le dépôt Git associé :

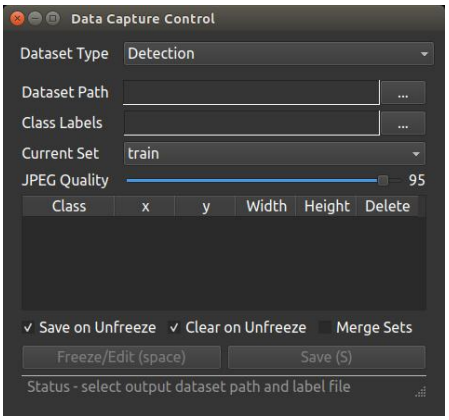
https://github.com/dusty-nv/jetson-inference

Pour développer une IA qui reconnaît des mégots de cigarette, nous devons tout d'abord l'entraîner. Pour comparer, notre IA fonctionne de la même façon que n'importe quel apprentissage : en répétant les entraînements pour mieux apprendre.

Pour faire ces entraînements, nous devons donc créer une base d'image de mégots de cigarette répertorié en 3 dossiers :

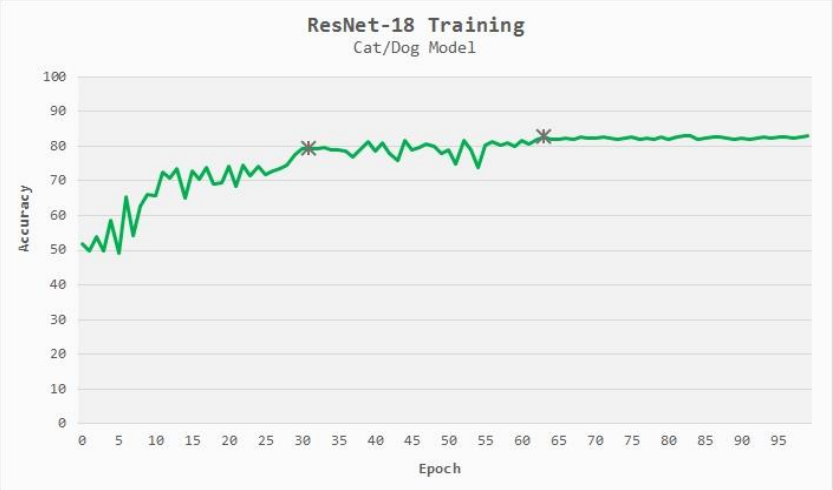
- **/train** : Ce sont les données qui vont entraîner notre modèle
- **/test** : Ce sont les données qui vont tester notre modèle
- **/val** : Ce sont les données qui vont valider notre modèle

Pour faciliter le travail, le git associé dispose d'une logiciel permettant de créer automatiquement les dossiers et d'y ranger les images prises directement via une Webcam :



Lorsque nous allons procéder à l'entraînement de notre modèle, l'IA va passer par chaque image de chaque dossier en commençant par le dossier **/train** pour s'entraîner, puis par le dossier **/test** pour se tester et enfin par le dossier **/val** pour se valider. Un passage dans l'ensemble des données est appelé **epoch**. A la fin d'un epoch, nous obtenons donc une IA capable de reconnaître un mégot de cigarette avec une certaine précision. Le nombre d'epoch ainsi que le nombre d'images a une incidence sur la précision obtenue.

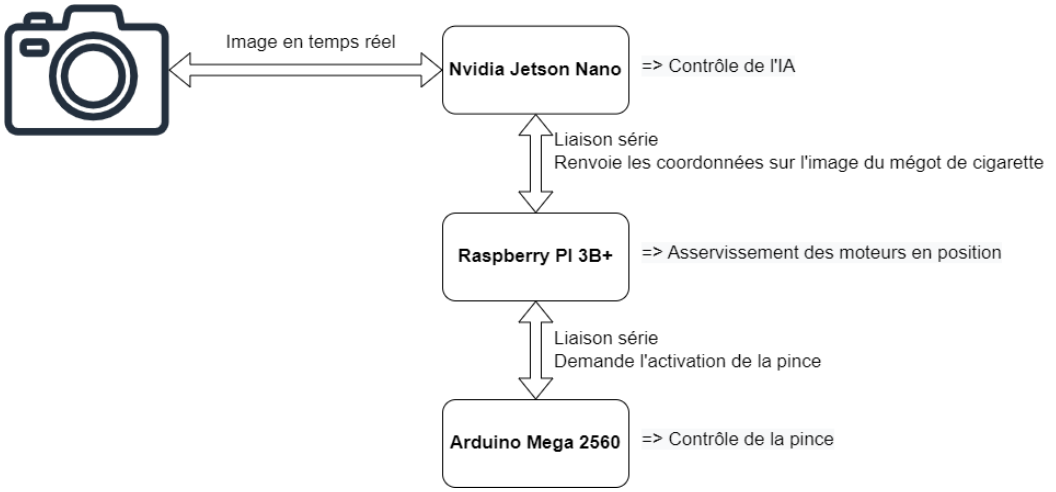
Le Git associé nous donne un exemple avec une base d'image de 5000 images de chien et de chat :



Nous remarquons donc que pour 5000 images, nous tendons vers une précision de 80% à partir de 30 epochs.

Après cette partie d'analyse, nous avons récolté des mégots de cigarette que nous allons utiliser pour la création de la base d'image puis nous allons créer notre modèle et le tester.

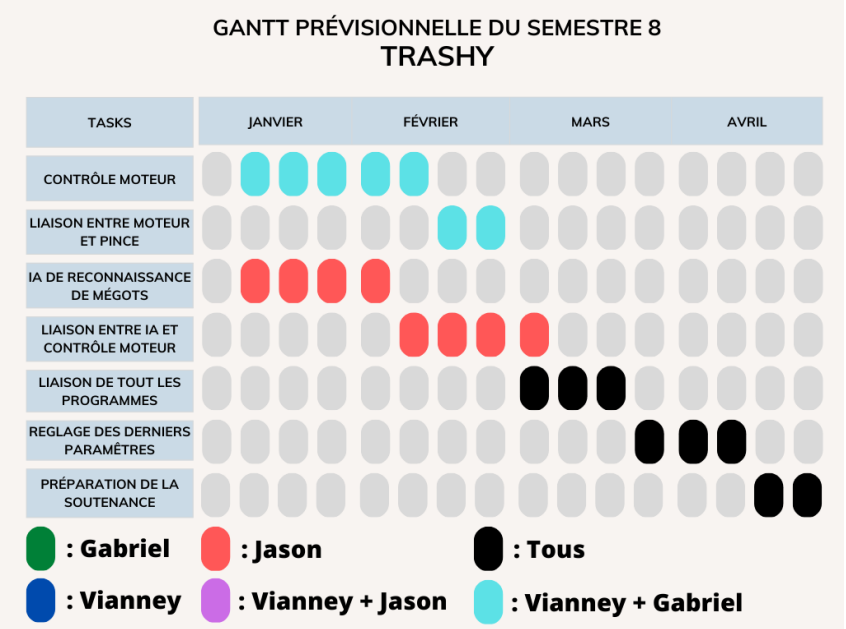
En parallèle de cette analyse, nous avons aussi réfléchi sur comment les différentes parties ainsi que les différents modules allaient communiquer entre eux :



Il est totalement possible de regrouper les fonctions de la Nvidia Jetson Nano et de la Raspberry sur le même module pour du gain de place mais nous restons sur cette configuration pouvoir travailler sur les différents fonctions en même temps

III. Ce qu'il nous reste à faire

Afin de mieux prévoir le semestre 8, nous avons fait l'inventaire des tâches à faire et nous avons réalisé le diagramme de Gantt ci-dessous :



Il nous reste donc le contrôle moteur ainsi que l'IA de reconnaissance de mégot à programmer et à faire la liaison entre tous les programmes. Nous nous laissons une marge de trois semaines pour régler les éventuelles imprévus et problèmes que nous pourrions rencontrer lors de notre projet.

Conclusion

Lors de ce semestre, nous avons pu voir les limites de notre méthode d'organisation et de planification. En effet, au début nous avions planifié un diagramme de Gantt pour ce semestre mais seulement 4 tâches ont été réussi sur 7.

Sur beaucoup de points, nous avons sous-estimé la difficulté des tâches et nous avons pû mieux nous organiser pour le semestre qui arrive afin d'avoir un projet fini à la fin de notre année.