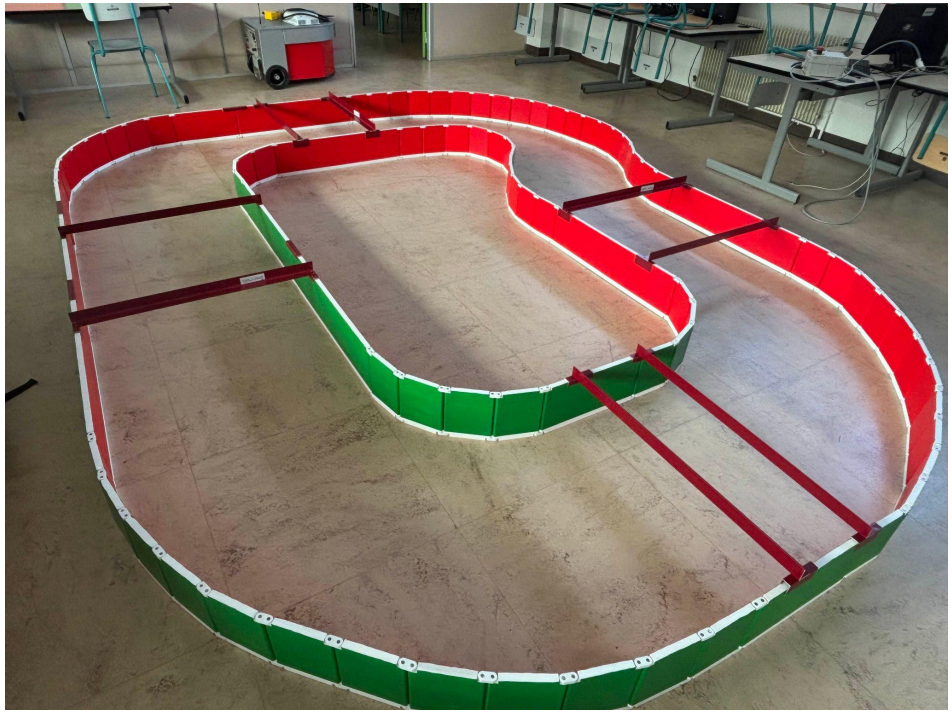
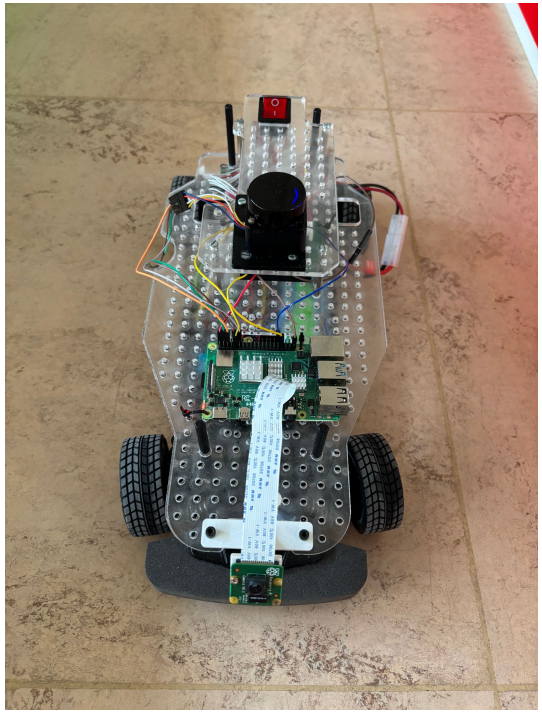
Revue Projet COVACIEL

Sommaire

- Présentation projet
- Analyse des classes
- Langage et version utilisé
- Problématique physique
- Diagramme au format UML
- Problème Actuel

Présentation projet





Expression des besoins

Objectifs

- Navigation autonome
- Détection et évitement des obstacles
- Interaction avec l'environnement

Besoins Fonctionnels et techniques:

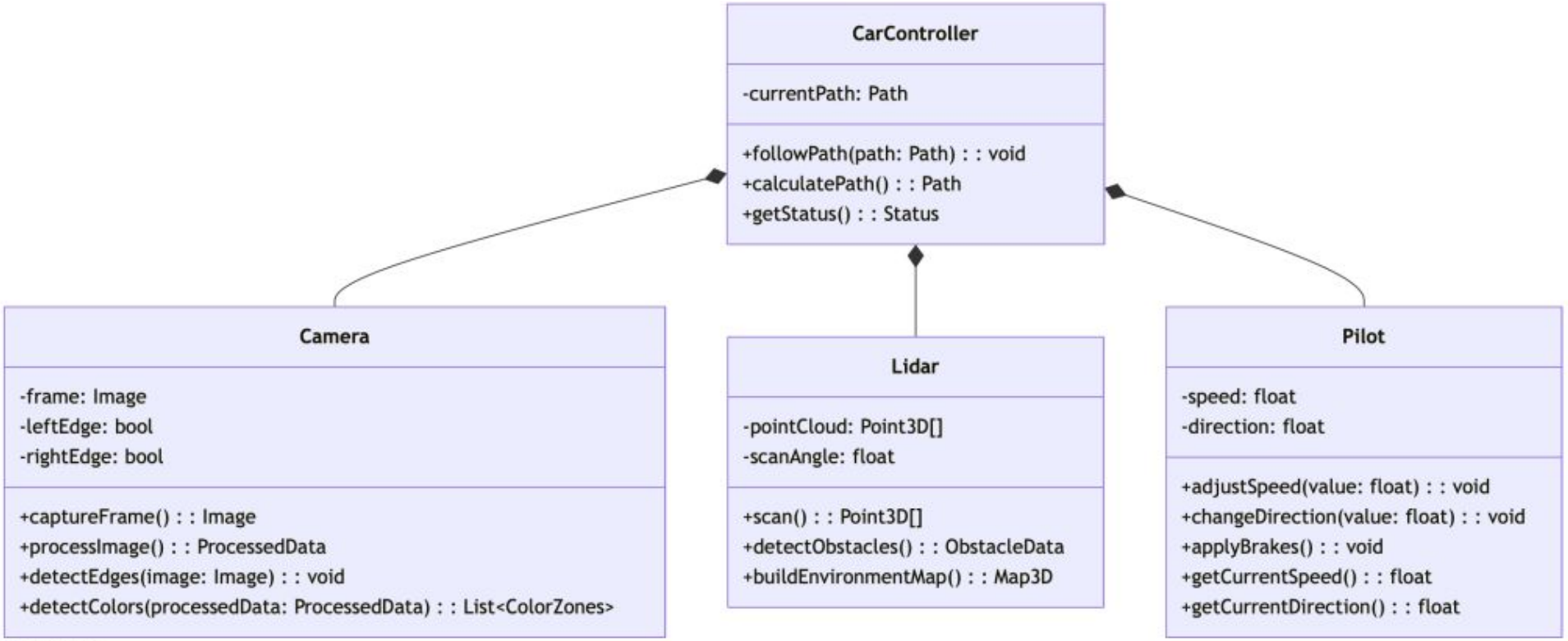
Matériel:

Unité de traitement: Rasberry Pi4 (4GB RAM), Carte microSD 32GB

Capteur : Caméra Rasberry Pi v2, Lidar RPlidar LD06

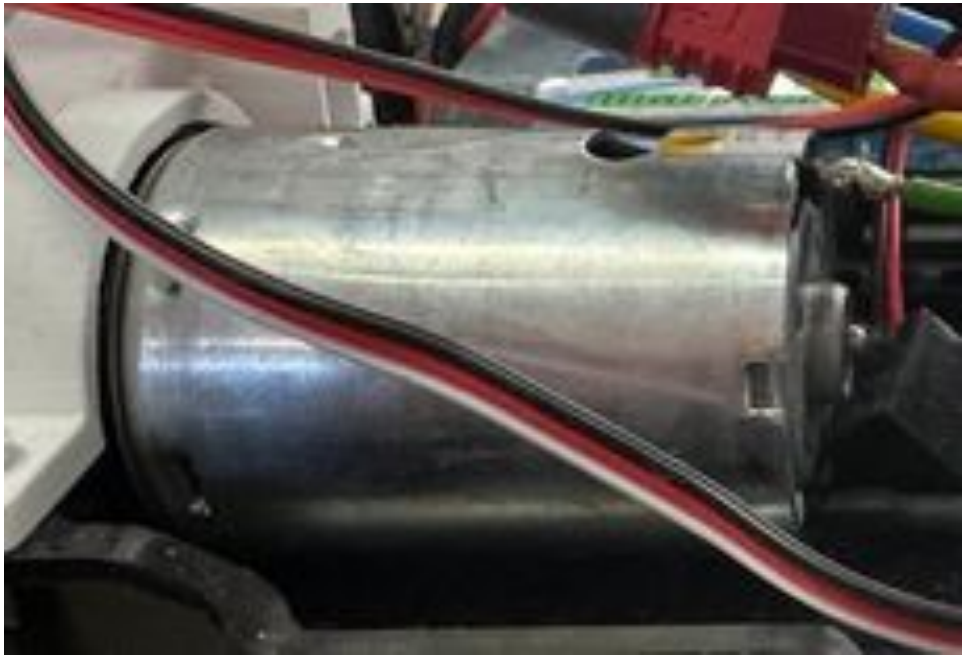
System d’exploitation : Rasberry Pi OS (64-bit)

Répartition des tâches



Langage et version

Présentation Moteur



Problématique physique

Spasme de vitesses : Augmentation et diminution soudaine de la vitesse

Inertie du moteur : changement brutal du rapport cyclique, le moteur n’a pas le temps d’atteindre progressivement sa nouvelle vitesse

Solution

Vérification du câblage : souder les câbles.

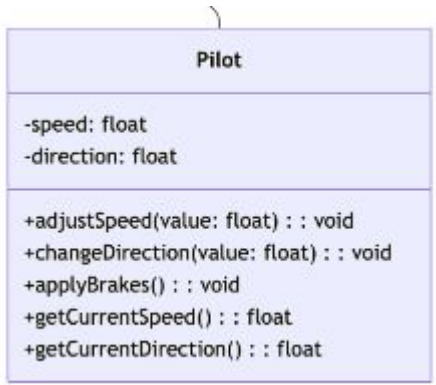
Mise en thread : Réduire les sources potentielles d'interférences logiciel.

Classe au format UML

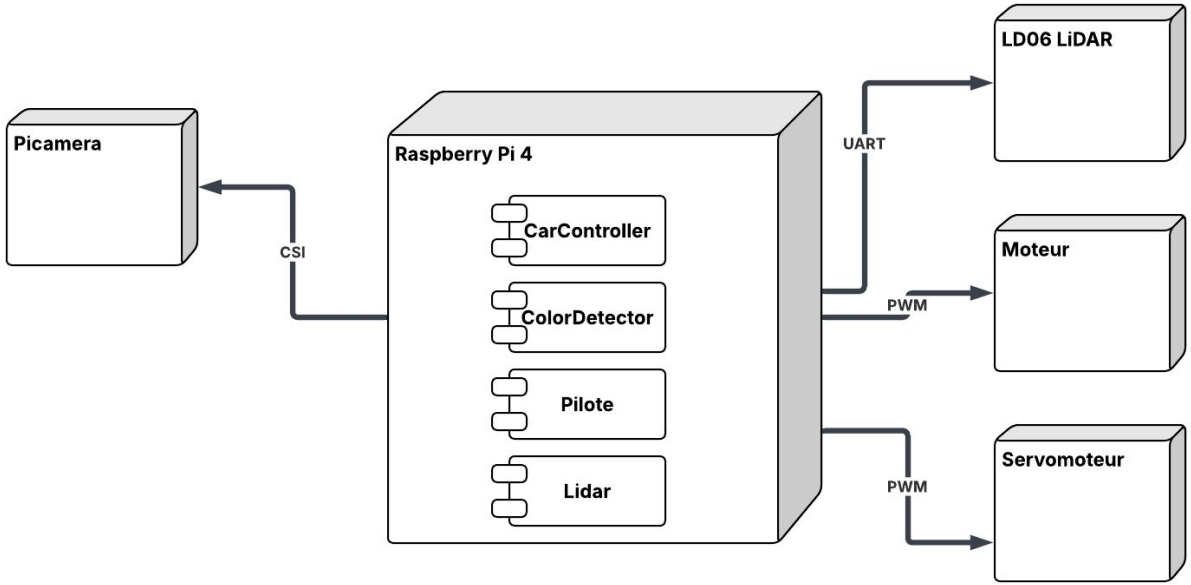
Pilote

-float speed
-float direction
-int branch_moteur
-int branch_direction
-int branch_fourche
-Lock lock
-Event update_event
-Thread pilote
-PWM pwm
-PWM dir

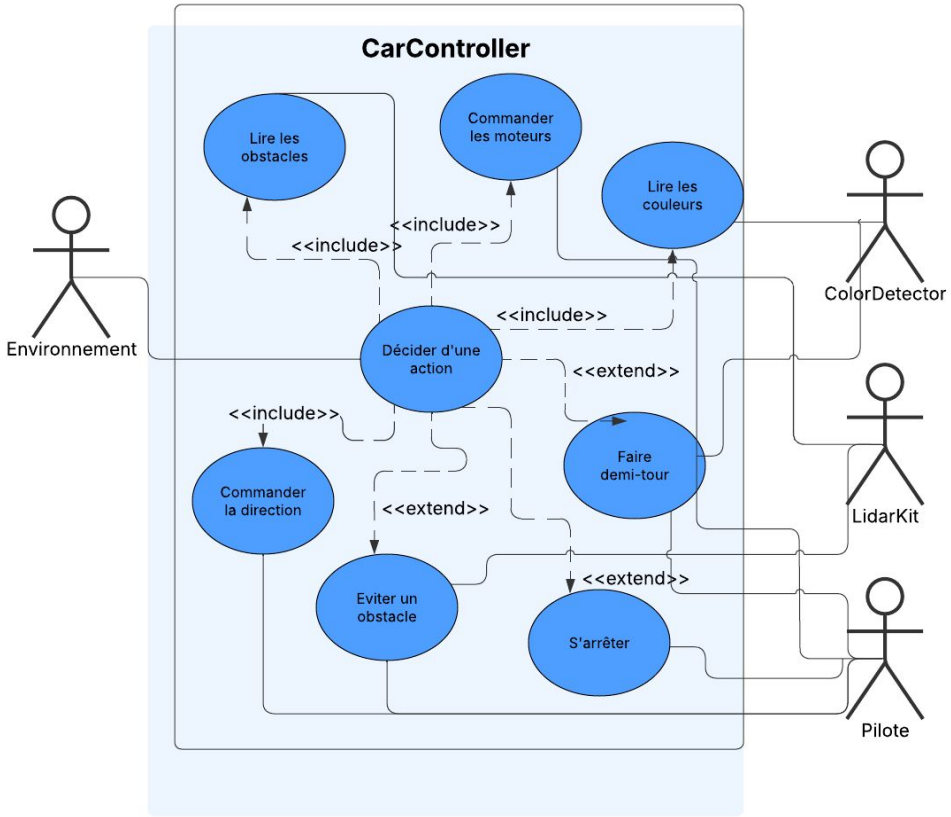
+**init**(speed, direction, branch_moteur, branch_direction, branch_fourche)
+changePilote()
+UpdateCar(ID, new_value)
+applyBrakes(entrer)
+getCurrentSpeed() : float
+getCurrentDirection() : float
+verificationEntrer(Control_car_input) : float
+calculerRapportCyclique(ID) : float
+genererSignalPWM(ID, rapportCyclique)
+stop()
+GetFourche()
+CalcPID(input, output) : float



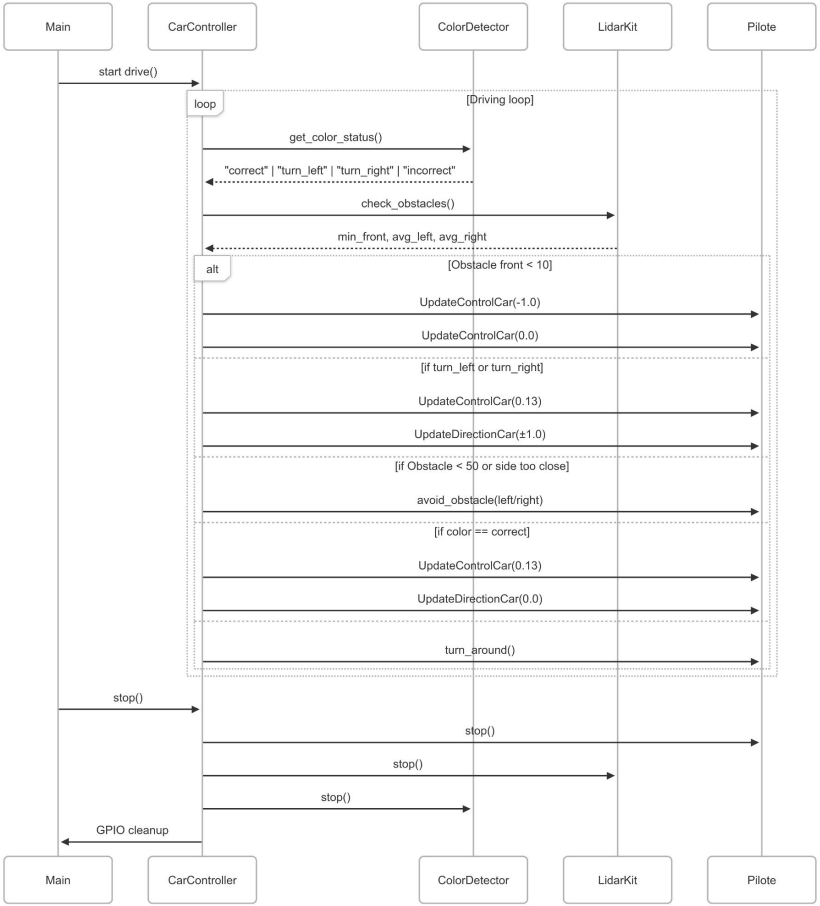
Déploiement au format UML



Cas d’utilisation au format UML



Séquence au format UML



Pilote classe

```python
def adjustSpeed(self):      You, il y a 2 semaines • Update Pilote class …
    # Methode lu par le thread qui s'occupe d'ajuster en boucle la vitesses
    global Control_car_input

    while running:
        self.speed = Pilote.verificationEntrer(Control_car_input) # Ajuste la valeur de la v
        rapportCyclique = Pilote.calculerRapportCyclique(self, 0) # Génère un rapport cycliq
        Pilote.genererSignalPWM(self, 0, rapportCyclique) # Envoie le resultat du rapport cy

    return
```

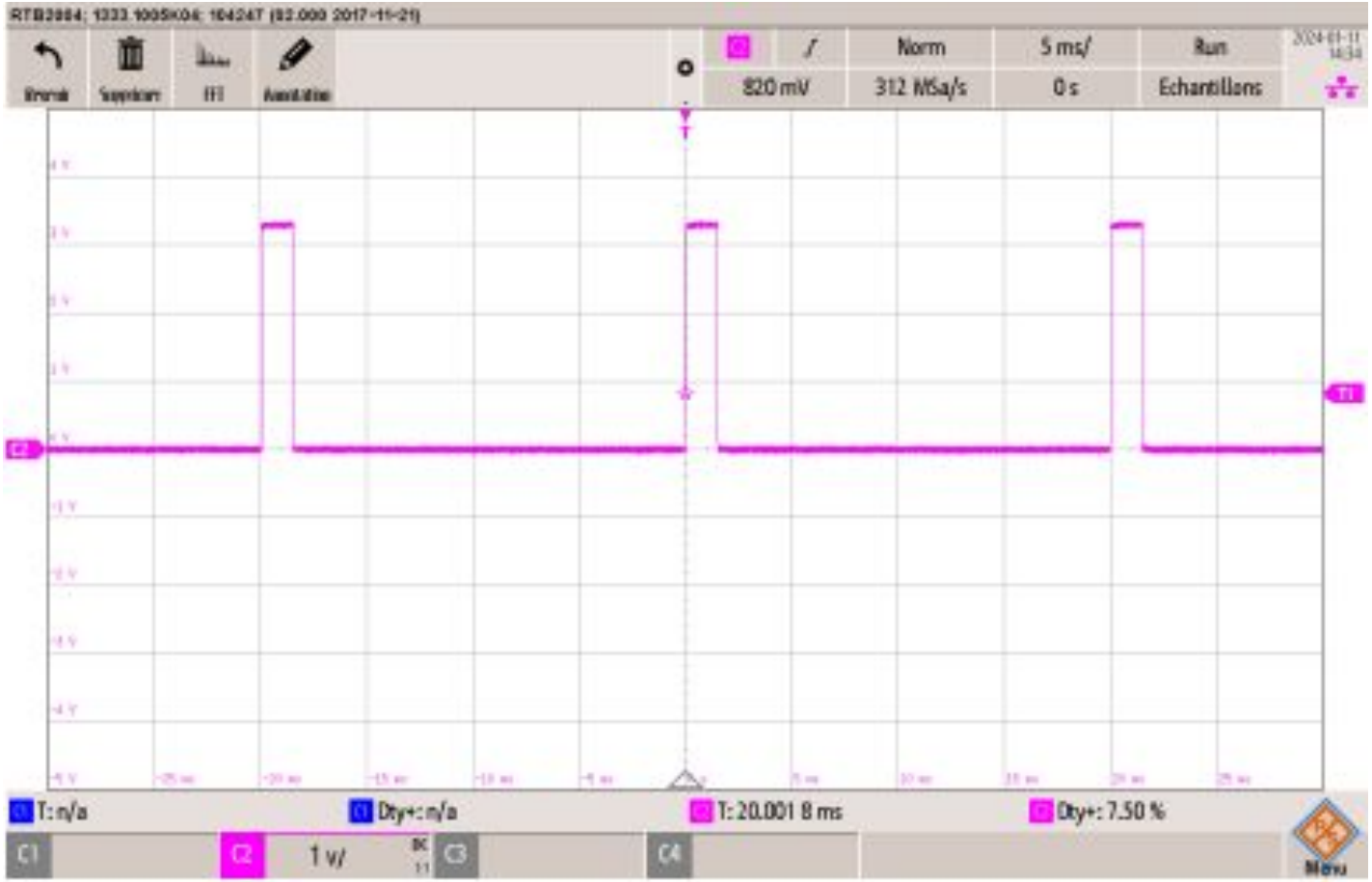
```python
def UpdateControlCar(self, new_value):
    """Méthode Pour effectuer la mise a jour de la vitesse de la voiture entre -1 et 1"""
    # Methode intermediaire pour ajuster la valeur de control_car_input
    global Control_car_input

    with self.lock:
        Control_car_input = new_value
```

Méthode de génération PWM

```python
def calculerRapportCyclique(self, ID):
    """Effectue le calcule pour le rapport cyclique moteur et direction"""
    periode = 20e-3  # Période de 20 ms (0.020 s)

    if ID == 0:
        speed = self.speed

        # Calcul du rapport cyclique pour la vitesse
        if speed >= 0:
            temps_haut = 1.5e-3 + speed * (2.0e-3 - 1.5e-3)  # Jusqu'à 2.0 ms
        else:
            temps_haut = 1.5e-3 + speed * (1.5e-3 - 1.3e-3)  # Jusqu'à 1.3 ms

        rapport_cyclique = (temps_haut / periode) * 100
        return rapport_cyclique

    elif ID == 1:
        direction = self.direction

        # Calcul du rapport cyclique pour la direction
        if direction == 0:
            temps_haut_direction = 1.38e-3  # Temps haut de 1.38 ms
        elif direction == 1:
            temps_haut_direction = 1.54e-3  # Temps haut de 1.54 ms
        elif direction == -1:
            temps_haut_direction = 1.22e-3  # Temps haut de 1.22 ms

        rapport_cyclique = (temps_haut_direction / periode) * 100
        return rapport_cyclique
```

```python
def genererSignalPWM(self, ID, rapportCyclique):
    """
    Utilise un signal PWM pour l'injecter soit dans :
    - le moteur (ID = 0)
    - la direction (ID = 1)
    """
    # Génère un signial pwm pour le moteur si l'ID est 0
    if ID == 0:
        self.pwm.ChangeDutyCycle(rapportCyclique)
    # Génère un signial pwm pour le la direction si l'ID est 1
    elif ID == 1:
        self.dir.ChangeDutyCycle(rapportCyclique)
```

Problème Actuel

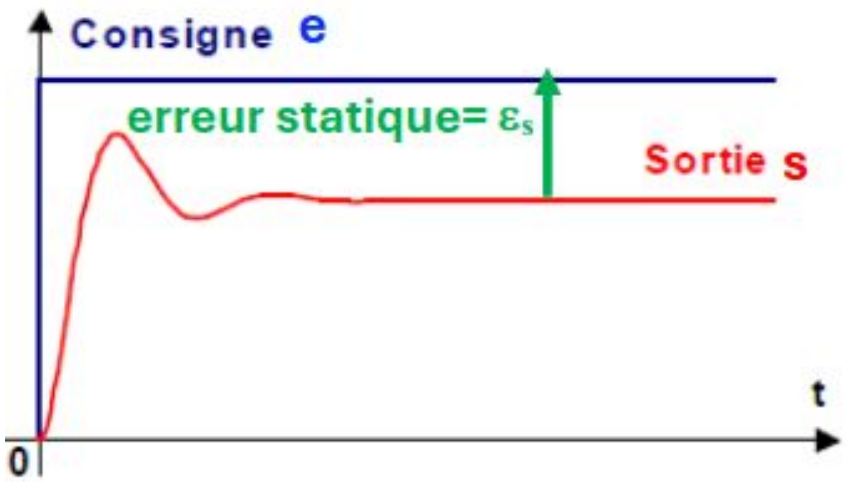


Révision des méthodes

```
def changePilote(self):
    # Methode lu par le thread qui s'occupe d'ajuster en boucle la direction
    global Control_direction_input ; global update_dir
    global Control_car_input ; global update_moteur

    while running:
        self.update_event.wait()  # Attente jusqu'à ce qu'une mise à jour soit demandé
        self.update_event.clear()

        if update_dir == True:
            self.direction = Pilote.verificationEntrer(self, Control_direction_input)
            rapportCyclique = Pilote.calculerRapportCyclique(self, 1) # Génère un rapp
            Pilote.genererSignalPWM(self, 1, rapportCyclique) # Envoie le resultat du
            update_dir = False

        if update_moteur == True:
            self.speed = Pilote.verificationEntrer(self, Control_car_input) # Ajuste l
            rapportCyclique = Pilote.calculerRapportCyclique(self, 0) # Génère un rapp
            Pilote.genererSignalPWM(self, 0, rapportCyclique) # Envoie le resultat du
            update_moteur == False

        else:
            continue      You, il y a 5 jours • Update
    return
```

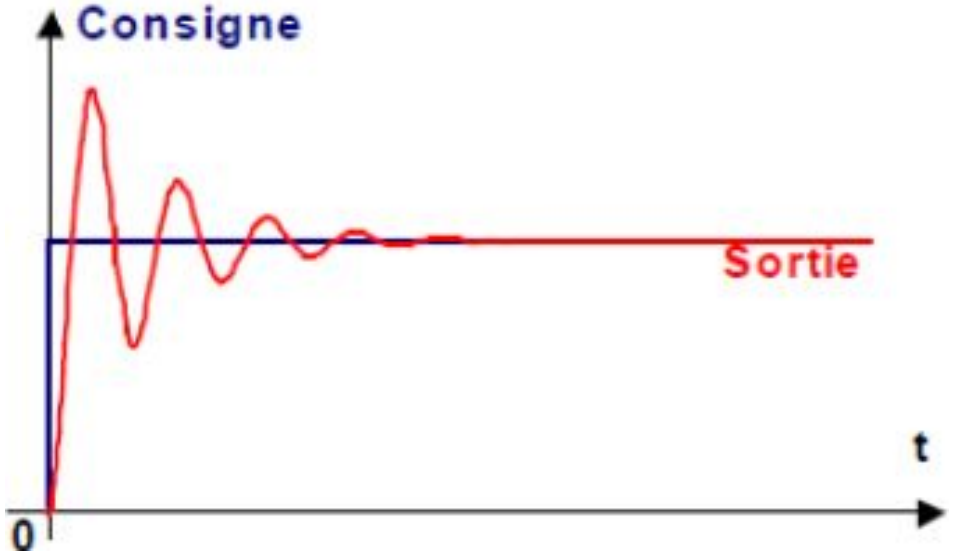
```python
def UpdateCar(self, ID, new_value):
    """Méthode Pour effectuer la mise a jour de la direction ou la vitesse moteur
    - le moteur (ID = 0)
    - la direction (ID = 1)
    """
    # Methode intermediaire pour ajuster la valeur de control_direction_input et c
    global Control_direction_input ; global update_dir
    global Control_car_input ; global update_moteur

    if ID == 1:
        with self.lock:
            Control_direction_input = new_value
        update_dir = True
    elif ID == 0:
        with self.lock:
            Control_car_input = new_value
        update_moteur = True
    else:
        return 1

    self.update_event.set()  # Réveille le thread
```

Utilité d’un PID

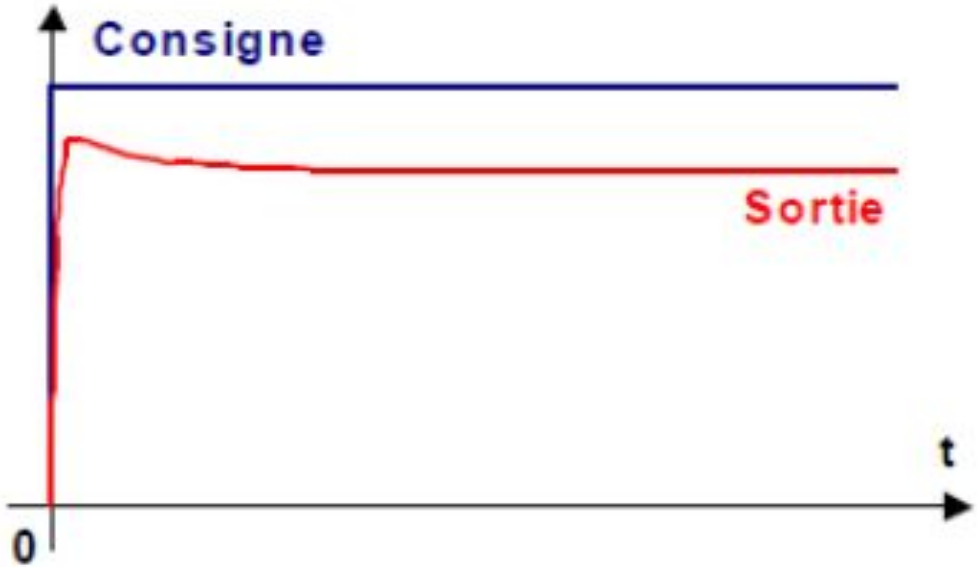


P

P I



P D



Méthode du calcule P

```
def CalcPID(self, input, output):
    # Calcule P
    gain = 10 # Kp
    e = input - output # Consigne moins sortie
    P = gain * e # gain * e (erreur)

    return P      You, il y a 6 secondes • Uncommitted changes
```

Conclusion