Revue Projet COVACIEL

Sommaire

- Introduction
- Présentation projet
- Expression des besoins
- Répartition des tâches
- Environnement de développement
- Présentation lidar
- Problématique physique liée au lidar
- Solution
- Diagramme de classe
- Pilote classe
- Méthode
- Probleme Actuelle
- Conclusion

Présentation projet

Expression des besoins

Objectifs

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

Besoins Fonctionnels et techniques:

Matériel:

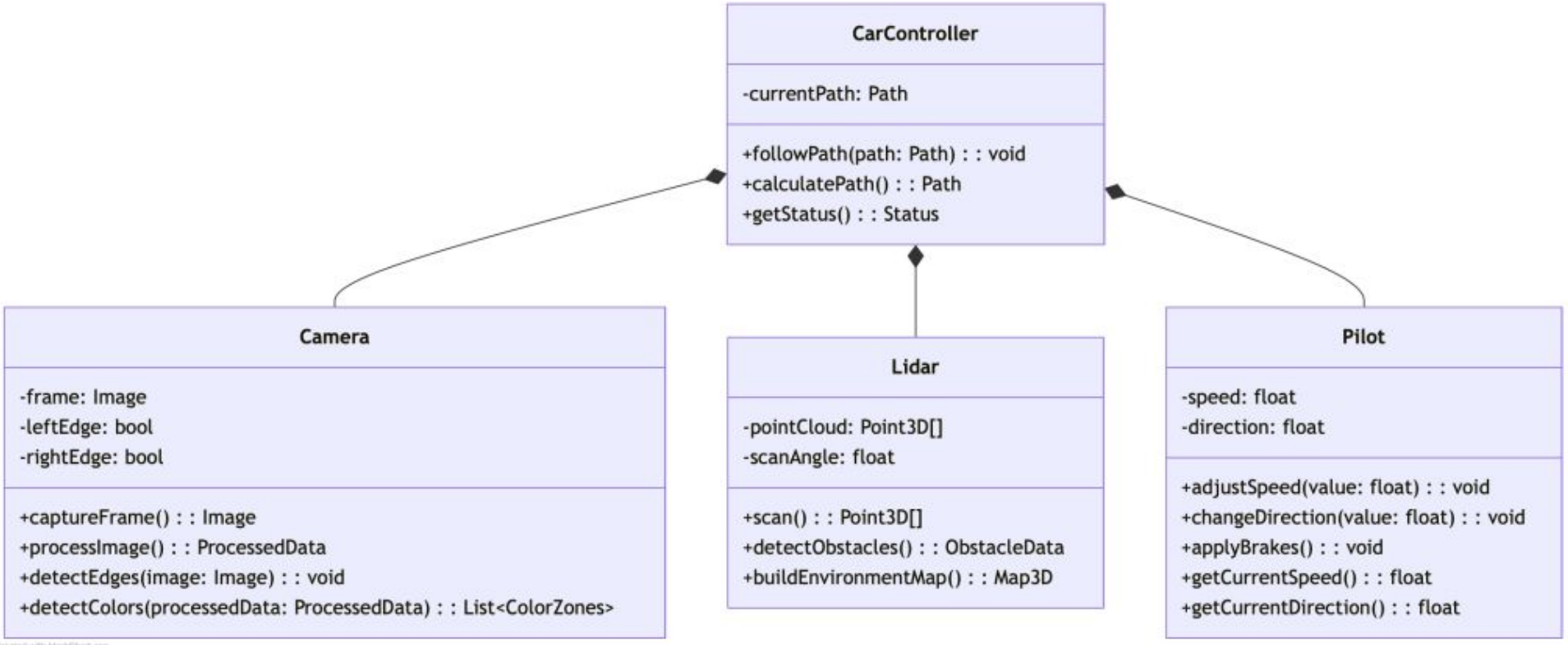
Unité de traitement: Rasberry Pi4 (4GB RAM), Carte microSD 32GB

Capteur : Caméra Rasberry Pi v2, Lidar RPlidar LD06

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

Outils : SSH

Répartition des tâches



Environnement de développement

Présentation lidar

Le Lidar LD06 est un capteur laser compact pour mesurer les distances avec précision. Il offre une portée de mesure jusqu'à 12 mètres et réalise des mesures à une fréquence de 4 500 Hz, permettant une détection rapide et fiable des obstacles. Grâce à son balayage à 360 degrés, il est capable de cartographier l'environnement en temps réel, ce qui le rend idéal pour des applications telles que la navigation autonome et l'évitement d'obstacles.

Problématique physique

Surfaces réfléchissantes ou transparentes : les objets type mirroir ou transparents peuvent perturber le signal laser,envoyant donc de mauvaise distance.

Réflexions multiples : Les signaux laser peuvent rebondir sur plusieurs surfaces avant de retourner au capteur, entraînant des mesures incorrectes.

Interférences lumineuses : Une forte lumière ambiante ou d'autres sources laser peuvent interférer avec le capteur, affectant la précision des mesures.

Solution

Calibration régulière : Effectuez des calibrations du capteur pour maintenir sa précision et compenser d'éventuelles dérives

Optimisation de l'environnement : Réduisez les sources potentielles d'interférences en contrôlant l'éclairage ambiant et en évitant les surfaces problématiques dans la zone de mesure.

Optimisation de l’utilisation du Lidar : Ajuster le rapport cyclique du signal PWM peut influencer la qualité et la quantité des données en modifiant la vitesse de rotation du LiDAR.

Diagramme de classe

Pilote

- speed: float
- direction: float
- branch_moteur: int
- branch_direction: int
- pwm: PWM

+ **init**(speed: float, direction: float, branch_moteur: int, branch_direction: int)
+ adjustSpeed(Control_car_input: float) : : void
+ changeDirection(Control_direction_input: float) : : void
+ applyBrakes(entrer: bool) : : bool
+ getCurrentSpeed() : : float
+ getCurrentDirection() : : float
+ verificationEntrer(Control_car_input: float) : : float
+ calculerRapportCyclique(): tuple(float, float)
+ genererSignalPWM(rapportCyclique: float) : : void

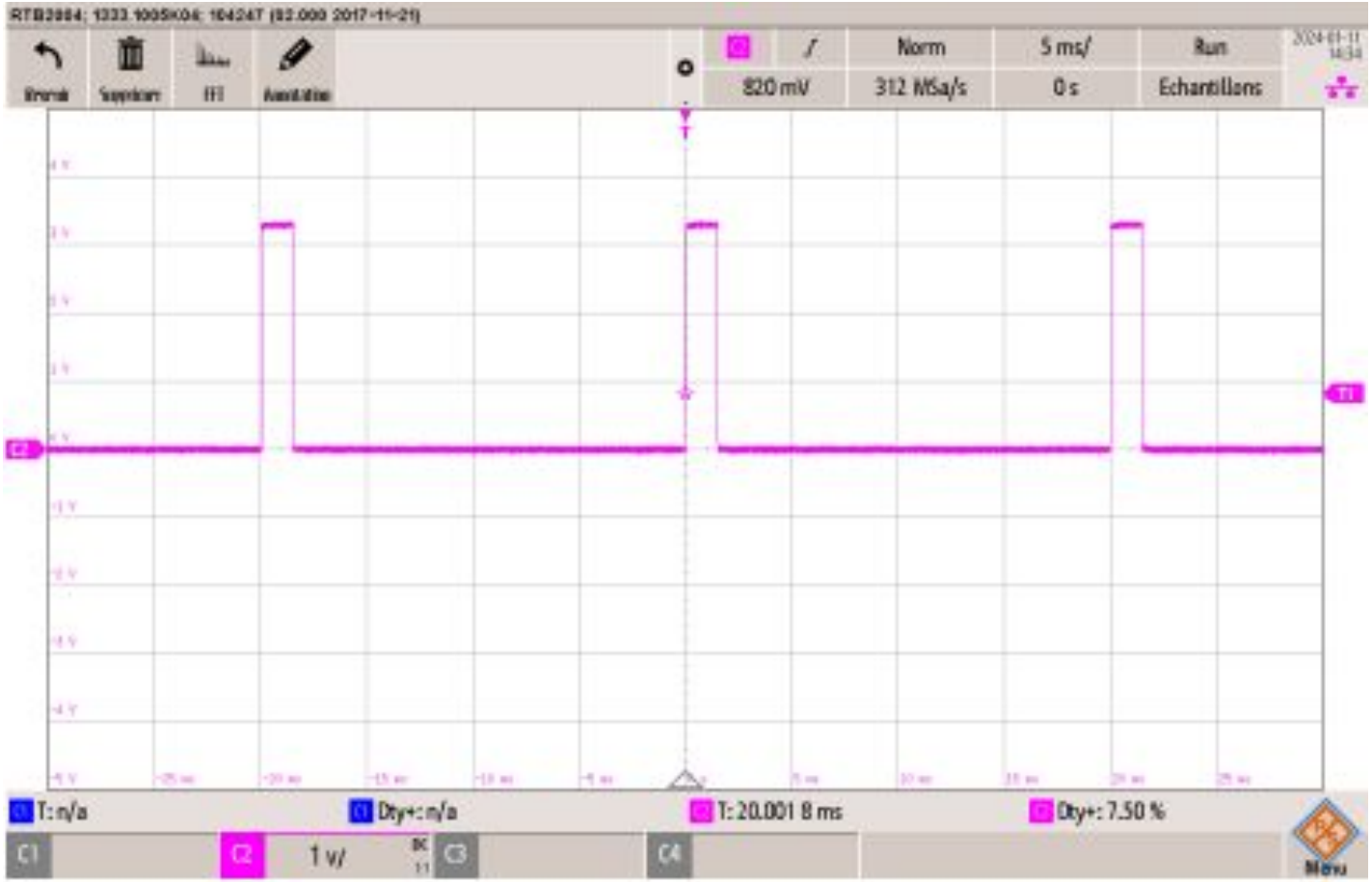
Pilote classe

```
def adjustSpeed(self, Control_car_input):
    # Faire un system de com entre le raspi et le resultat de notre decision
        You, il y a 2 mois • Update
    self.speed = Pilote.verificationEntrer(Control_car_input) #Ajuste la valeur de la vitesse
    rapportCyclique, temps_haut = Pilote.calculerRapportCyclique(self)
    Pilote.genererSignalPWM(self, rapportCyclique)
    print(f"Ajustement de speed a : {self.speed} \n Rapport cyclique : {rapportCyclique}%\n

    return
```

Méthode de génération PWM

```
def calculerRapportCyclique(self):
    speed = self.speed
    direction = self.direction
    periode = 20e-3  # Période de 20 ms (0.020 s)

    if speed >= 0 or direction >= 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, temps_haut


def genererSignalPWM(self, rapportCyclique):
    sleep(0.05)
    self.pwm.ChangeDutyCycle(rapportCyclique)
    return
```

Probleme Actuelle



Conclusion