Quentin BUTZBACH, Hubert-Emmanuel DE LAFORCADE, Louis LANOY, Ethan PUYAUBREAU, Sriviboosan SRIPATHY

Course de Voitures Autonomes :

Présentation des voitures de l'Institut Villebon Georges Charpak

Institut Villebon Georges Charpak

université PARIS-SACLAY

Présentation du véhicule

Nous disposons de deux voitures, dénommées Alpha et Omega, basées sur un même châssis RC (TT-02).

Le guidage autonome est géré par deux algorithmes différents pour les deux voitures :

- Algorithme déterministe
- Réseau Neuronal

Ces algorithmes tournent sur la Raspberry Pi.

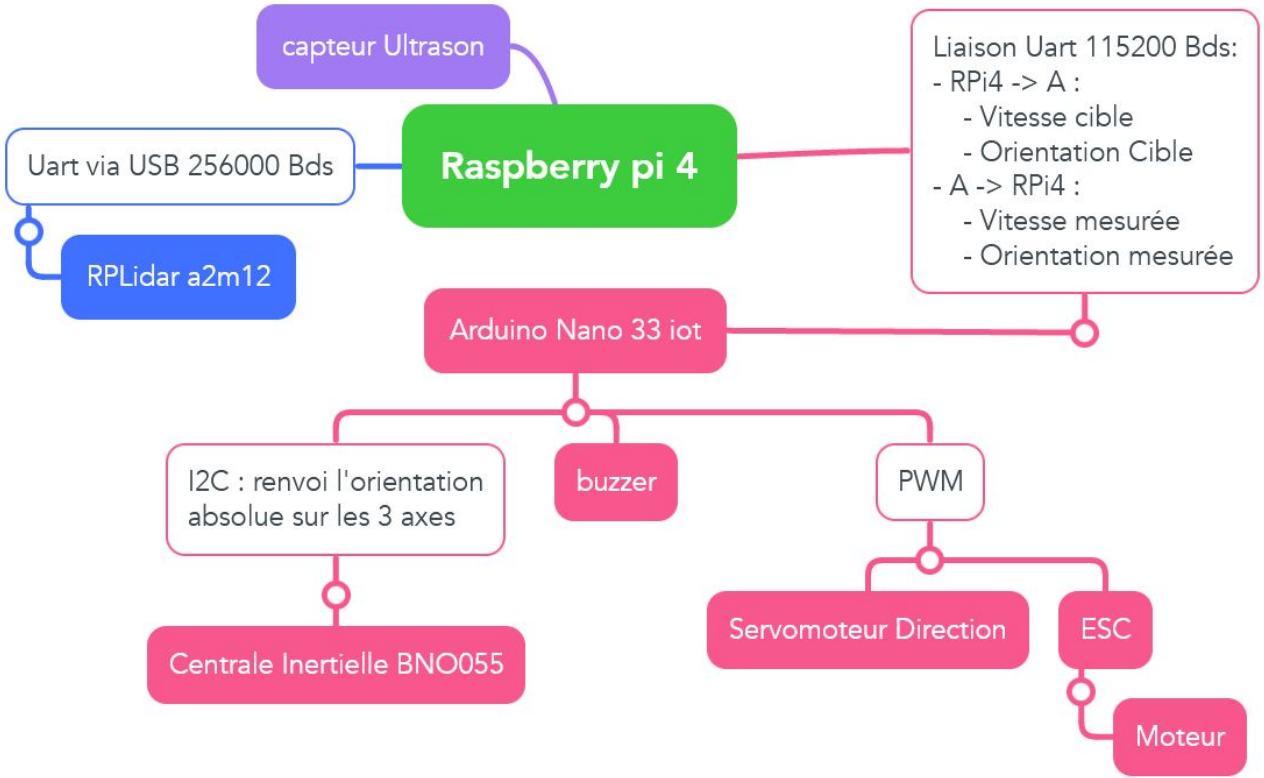


Schéma synoptique (valable pour les deux voitures)

Arduino et Asservissement de vitesse

L'asservissement de vitesse se fait sur l'arduino à l'aide de la librairie AutoPID de Ryan Downing :

- La mesure de vitesse se fait à partir de la fourche, en utilisant les Hardware Interrupt de l'arduino. Une soudure a dû être faite, le pin original ne permettant pas l'hardware interrupt.
- L'arduino mesure le temps entre deux fronts montants de la fourche. Elle peut en déduire la vitesse de rotation de L'axe et ainsi la vitesse de la voiture.
- Les vitesses irréalistes (trop grandes) sont ignorées.
- La vitesse utilisée par la boucle PID est une moyenne des vitesses mesurées au cours du cycle (20ms).
- La boucle est mise en pause quand le véhicule est à l'arrêt.

L'asservissement de la direction se fait en utilisant le même outil, AutoPID :

- La boucle prend en mesure la différence entre le cap demandé par la Raspberry Pi et l'orientation fournie par la centrale inertielle. Cela permet de garder le contrôle sur les cas spéciaux.
- La boucle prend en consigne 0.

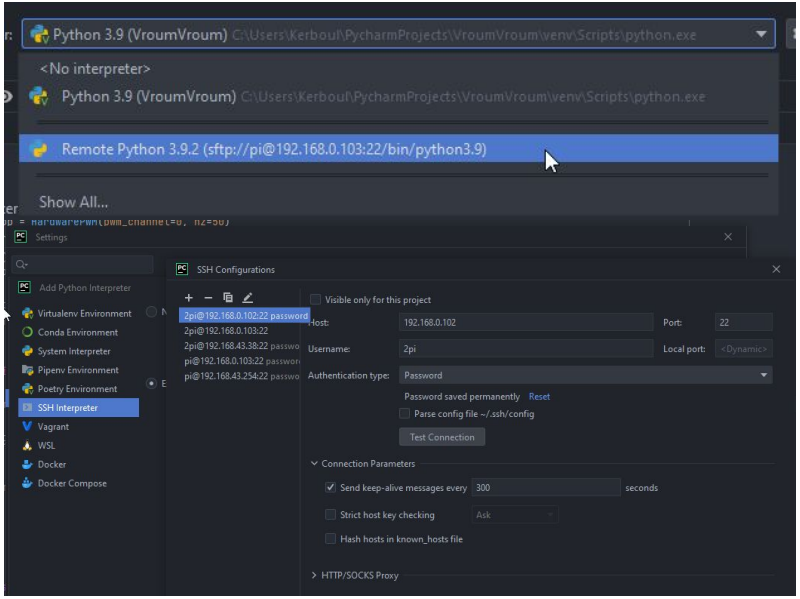
L'arduino communique avec la Raspberry pi via la liaison Uart :

- l'arduino reçoit une vitesse cible ainsi qu'un cap de direction. dans le cadre de l'algorithme par secteur, le cap est une direction absolue tandis-que dans le cadre du réseau neuronal, le cap est analogue à un taux de rotation. Le code est adapté manuellement pour les deux cas d'usage.
- l'arduino renvoi la vitesse mesuré et le cap mesuré.

Le Buzzer sert d'alerte sonore en cas de problème.

Interaction et simulation

Commande et interpréteur à distance



On a utilisé PyCharm afin d'établir un lien SSH, les avantages étant :

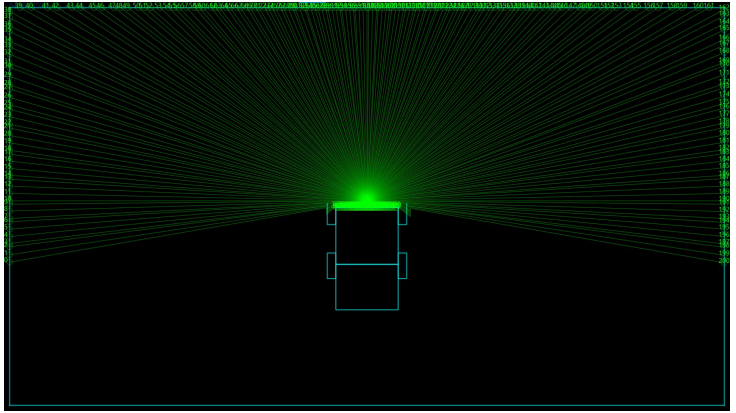
- Synchronisation automatique des fichiers locaux (sur ordi) avec ceux sur la Raspberry Pi
- Interaction directe avec la voiture sans besoin de VNC
- Importe et installe automatiquement les librairies utilisées sur la voiture (python doit être installé sur la Pi)

[Lien vers l'explication utilisée pour configurer un interpréteur SSH (jetbrains.org)]

Intégration du programme secteurs

```
donnees_lidar_mm = aquisition_donnees_lidar()
Sectors = []

Len = len(donnees_lidar_mm)
Num = 0

for i in range(Len):
    Data = donnees_lidar_mm[i]
    if ma.isinf(Data) == False:
        Num += int(Data*1000)
    Val = i%10      # Toutes les 10 valeurs
    if Val == 0:
        Sectors.append(Num) # on stocke la som
        Num = 0

Sectors.append(Num)     # Même si le dernier s
```

```
Len = len(Sectors)
Middle = Len/2
AngleStep = maxangle/Middle
SpeedStep = maxSpeed/Len
Angle = 0
Speed = 0

for index in range(Len):
    Value = Sectors[i]
    #
    if Value < 30:      # Si un secteur est inférieur à 3
        if (index-Middle) < 0:  # En fonction de l'index
            Angle += AngleStep      # On incrémente
        if (index-Middle) > 0:      # Ou
            Angle -= AngleStep      # On Décrémente l'ang

    if Value > 80:
        Speed += SpeedStep      # Pour chaque Secteur ou
    elif Value > 40:
        Speed += SpeedStep/2      # Pour chaue Secteur ou
```



Représentation graphique du scan lidar dans l'algo "secteur" pour une future implémentation dans Webots

Apprentissage par renforcement

```python
def main():
        env = WebotsGymEnvironment()
        print("environnement créé")
        check_env(env)
        print("vérifiaction de l'environnement")

        # Model definition
        model = PPO(policy='MlpPolicy',
                                    env=env,
                                    learning_rate=5e-4,
                                    verbose=1,
                                    device='cpu')
```

```python
# Step function
def step(self, action):
    self.drive(action)
    obs = self.get_observation()
    super().step()
    reward, done = self.get_reward(obs)
    return obs, reward, done, {}
```

- Intégration de Gym dans Webots
- Usage d'un agent PPO comme politique d'apprentissage
- Fonction de récompense favorisant l'éloignement des murs et punissant les crashs
- Entraînement avec d'autres véhicules obstacles roulant sur le circuit

```python
for _ in range(10000):
        action, _ = model.predict(obs, deterministic=True)
        obs, reward, done, _ = env.step(action)
        c += reward
        if done:
                obs = env.reset()
```